

FAIR DIVISION OF INDIVISIBLES:
ON THE COMPUTABILITY OF MAXIMIN SHARE (MMS) ALLOCATIONS

BY

SETAREH TAKI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Industrial Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2022

Urbana, Illinois

Doctoral Committee:

Assistant Professor Jugal Garg, Chair, Director of Research
Professor Chandra Chekuri
Professor Rakesh Nagi
Associate Professor Karthekeyan Chandrasekaran
Assistant Professor Ruta Mehta

Abstract

Fair division is a fundamental problem in various multi-agent settings, where the goal is to divide a set of resources among agents in a *fair* manner. This thesis studies the case where m indivisible items need to be divided among n agents with additive valuations using the popular fairness notion of *maximin share* (MMS). The maximin value of an agent is the largest value she can guarantee herself if she is allowed to choose a partition of the items into n bundles (one for each agent), on the condition that she receives her least preferred bundle. An MMS allocation provides each agent a bundle worth at least her maximin share. While it is known that such an allocation need not exist [45, 39], a series of remarkable works [45, 40, 6, 12, 28] provided approximation algorithms for a $\frac{2}{3}$ -MMS allocation in which each agent receives a bundle worth at least $\frac{2}{3}$ times her maximin share. Later, Ghodsi et al. [31] showed the existence of a $\frac{3}{4}$ -MMS allocation and a PTAS to find a $(\frac{3}{4} - \epsilon)$ -MMS allocation for an $\epsilon > 0$. Most of the previous works utilize intricate algorithms and require agents' approximate MMS values, which are computationally expensive to obtain.

The focus of this thesis is the classic setting of the MMS problem where all items are assumed to be goods (positively valued). The first result in this setting is an alternative $\frac{2}{3}$ -MMS allocation that offers a simple algorithm and straightforward analysis. In contrast to other algorithms, the approach allows for a simple and intuitive understanding of why it works.

Using the intuition built by the simple $\frac{2}{3}$ -MMS algorithm, we develop a new approach that gives a simple algorithm for showing the existence of a $\frac{3}{4}$ -MMS allocation. Furthermore, the approach is powerful enough to be easily extended in two directions: First, we get a strongly polynomial time algorithm to find a $\frac{3}{4}$ -MMS allocation, where we do not need to approximate the MMS values at all. Second, we show that a $(\frac{3}{4} + \frac{1}{12n})$ -MMS allocation always exists. This considerably improves the approximation guarantee, most notably for small n . We note that $\frac{3}{4}$ was the best factor known for $n > 4$. To this day, this is the state of the art approximation factor for the MMS problem.

Besides the classic setting of MMS, this thesis also covers results on the generalization of the MMS problem. The first aspect is studying the Groupwise MMS (GMMS) problem, which is a stronger notion than MMS. An allocation is GMMS if every sub-allocation is also an MMS allocation. We show an algorithm that obtains $\frac{4}{7}$ -GMMS allocation after donating at most n items (that were intended to be allocated).

Another approach to generalize the MMS problem is to study the MMS problem as mixed manna. In mixed manna, an item can be a good for some agents and a chore (that causes disutility) for others. Specifically, a PTAS to obtain MMS value is investigated. Such PTAS is known for goods (and chores) manna and plays a substantial role in obtaining approximate MMS allocation in several works. In contrast to goods (and chores) manna, for the mixed manna, a recent result by [38] showed that finding even an approximate MMS value of an agent up to any approximation factor in $(0, 1]$ is NP-hard for general instances. In this thesis, the hardness result is complemented by considering two restricted settings and obtaining a PTAS for each setting. The first restricted setting is when the following conditions hold: (i) the number of agents is a constant, and (ii)

for every agent, her absolute value for all items is at least a constant factor of her total (absolute) value for all goods *or* all chores. The second restricted setting is when the absolute value of MMS is at least $1/\rho$ times either the total value of all goods or the total cost of all chores, for some constant $\rho \geq 1$.

To Rou, Pedar, and Maman.

Acknowledgments

First and foremost, I am incredibly grateful to my Ph.D. advisor, Prof. Jugal Garg, for all his guidance, support, and patience during my doctoral degree. I would not be a researcher today if not for his persuasion and belief in me.

I was fortunate to work with Prof. Ruta Mehta. During our collaboration, I was provided with the supervision of a brilliant mind and care of a beautiful heart. I am forever thankful to her.

I would like to thank my parents, who always encouraged me to follow my dreams and have been the most incredible support that someone could wish for.

Finally, I also express my gratitude to my husband for being the best teammate in life and teaching me to be stronger and to believe in myself. Having him on my side is the most incredible grace of my life.

Table of contents

Chapter 1	Introduction	1
1.1	The Classic MMS Problem	2
1.2	Groupwise Maximin Share (GMMS)	4
1.3	MMS values in Mixed Manna	4
1.4	Further Related Work	6
Chapter 2	Preliminaries and A Simple Greedy Algorithm	7
2.1	Introduction	7
2.2	The MMS Problem	7
2.3	Properties of Maximin Share	8
2.4	A 2/3-MMS Approximation	12
2.5	Discussion	16
Chapter 3	A 3/4-MMS Algorithm	18
3.1	Introduction	18
3.2	Existence of 3/4-MMS Allocation	19
3.3	Algorithm for 3/4-MMS Allocation	30
3.4	Existence of $(3/4 + 1/12n)$ -MMS Allocation	35
3.5	Conclusions	39
3.6	Missing Proofs	40
Chapter 4	4/7-GMMS Allocation After Donating Some Items	44
4.1	Introduction	44
4.2	Problem Statement and Notations	44
4.3	Summary of the 1/2-GMMS Algorithm	44
4.4	4/7-GMMS Allocation	46
4.5	Basics of the Algorithm	47
4.6	Algorithm Description	48
4.7	Conclusions	52
Chapter 5	MMS Value in Mixed Manna with Restricted Valuations and Constant n	55
5.1	Introduction	55
5.2	Problem Definition and Notations	55
5.3	Finding MMS values of Agents When $\mu \geq 0$	57
5.4	Finding MMS values of Agents When $\mu < 0$	62
5.5	Conclusions	63
Chapter 6	Bounded MMS Value in Mixed Manna	64
6.1	Introduction	64
6.2	Preliminaries and Notation	66
6.3	Algorithm for B-MMS when $\mu \geq 0$	66
6.4	Algorithm for B-MMS when $\mu < 0$	74
6.5	Conclusions	83
References		84

Chapter 1

Introduction

Fair division is a fundamental problem that arises naturally in various multi-agent systems [46, 21, 47, 44, 26, 23, 31], for example, school seats assignment, spectrum allocation, air traffic management, and allocating computing resources on a network. In this problem the goal is to divide a set of resources among agents in a *fair* manner. This has been a subject of intense study since the seminal work of Steinhaus [46] where he introduced the cake-cutting problem for $n > 2$ agents: Given a heterogeneous (divisible) cake and a set of agents with different valuation functions, the problem is to find a *fair* allocation. The two most well-studied notions of fairness are: 1) Envy-freeness, where each agent prefers her own share of cake over any other agents' share, and 2) Proportionality, where each agent receives a share that is worth at least $\frac{1}{n}$ of her value for the entire cake.

In this thesis, we study the *discrete* fair division problem where m *indivisible* items need to be divided among n agents with additive valuations. For this setting, no algorithm can provide envy-freeness or proportionality, in general. E.g., consider allocating a single item among $n > 1$ agents. This necessitated an alternate concept of fairness. Budish [23] introduced an intriguing option called *maximin share*, which has attracted a lot of attention [45, 39, 31, 28, 40, 12, 27, 6]. The idea is a straightforward generalization of the popular *cut and choose protocol* in the cake-cutting problem and a natural relaxation of proportionality. Suppose we ask an agent i to partition the items into n bundles (one for each agent), with the condition that the other $n - 1$ agents get to choose a bundle before her. In the worst case, i receives the least preferred bundle. Clearly, in such a situation, the agent will choose a partition that maximizes the value of her least preferred bundle. This maximum possible value is called i 's maximin share (MMS) value. In fact, when all agents have identical valuations, i cannot guarantee more than the MMS value. MMS value of an agent is formally defined in Definition 2.1 of Chapter 2. Denote the MMS value of the agent i by μ_i or simply μ if agents are identical.

MMS problem has been studied in various settings: *good manna* where all items are considered to be goods (positively valued) by all agents, *chore manna* where all items are assumed to be chores (negatively valued by all agents), and *mixed manna* where each item can be positively valued by some agents, and negatively valued by some. In previous works, good manna is the most popular setting for the MMS problem. Therefore, in this thesis, we call it the *classic setting*.

More general versions of MMS problem have also been investigated. One generalization, defined in [14], is the Group-wise MMS (GMMS) problem. An allocation is called GMMS, if for any subset of the agents, their corresponding sub-allocation, only considering the agents and items in that sub-allocation, is also an MMS

allocation.

This thesis has three main parts. The classic MMS problem which is the main focus (Chapters 2 and 3), GMMS problem (Chapter 4), and MMS in mixed manna (Chapters 5 and 6). In the following sections an overview of the results in this thesis is presented.

1.1 The Classic MMS Problem

Consider the problem of finding fair allocation of a set M of m indivisible items among a set N of n agents with additive valuations, using the popular notion of maximin share (MMS) as our measure of fairness. Let $v_{ij} \geq 0$ denote agent i 's value for item j , and i 's valuation of any bundle $S \subseteq M$ of items is given by $v_i(S) = \sum_{j \in S} v_{ij}$. Let $V = (v_1, \dots, v_n)$ denote the set of all valuation functions.

Each agent's MMS value is a specific objective that gives her an intuitive measure of the fairness of an allocation. For example, Gates et al. [30] showed that in real-life experiments maximin metric is preferred by participating agents over others. This raises a natural question: Is there an allocation where each agent receives a bundle worth at least her MMS value? An allocation satisfying this property is said to be maximin share allocation (MMS allocation), and if it exists, it provides strong fairness guarantees to each individual agent. However, Procaccia and Wang [45], through a clever counter-example, showed that MMS allocation might not exist but a $\frac{2}{3}$ -MMS allocation always exists, i.e., an allocation where each agent receives a bundle worth at least $\frac{2}{3}$ of their MMS value. Later, Ghodsi et al. [31] improved the factor by showing the existence of a $\frac{3}{4}$ -MMS allocation using a sophisticated technique with a very challenging analysis.

One important note is that most of the previous work are primarily existential results that do not provide any efficient algorithm to find such an allocation. The main issue in these techniques is that they require knowing the agents' MMS values. The problem of finding the MMS value of an agent is NP-hard,¹ but a polynomial-time approximation scheme (PTAS) exists [48]. Theoretically, one can use PTAS to find a $(\frac{3}{4} - \epsilon)$ -MMS allocation for an $\epsilon > 0$ in polynomial time. However, for practical purposes, such algorithms are not very useful for small ϵ . Hence, finding an efficient algorithm to compute a $\frac{3}{4}$ -MMS allocation remained open.

To summarize, in this setting, agents have non-negative valuations over the items and the valuation functions of the agents are assumed to be additive. The main questions to answer in this setting are: What is the highest value α for which an α -MMS allocation always exists? How to efficiently find such an allocation? This question is usually pursued from two different angles. First, *existence*: what is the highest factor α for which the α -MMS allocation always exists. Second, *algorithm*: can we design an efficient algorithm to obtain an α -MMS allocation? In the next three subsections we will discuss our results for these questions.

1.1.1 A Simple Algorithm

Chapter 2 contains most of the fundamental concepts which are extensively used in all later chapters. Also, as the first result, in Chapter 2, an algorithm to find a $\frac{2}{3}$ -MMS allocation for agents with additive valuations is presented. The approach combines the insights of [19, 12] with concepts developed in [31] to obtain an algorithm that is both simple to implement and analyze. Similar to the algorithm proposed in [31], our algorithm consists of two phases: *matching* and *bag filling*. However, unlike [31], the phases in our work are much simpler, and we do not need to compute agents' MMS values. Bag filling is a simple, greedy method to

¹Observe that the partition problem reduces to the MMS value problem with $n = 2$.

allocate ‘low-value’ items: We start with an empty bag and keep adding one item at a time. If the value of the bag is worth at least $\frac{2}{3}$ of any agent’s MMS, then we assign the bag to that agent, picking an arbitrary agent when there are more than one satisfying this condition. It can be shown that bag filling provides a $\frac{2}{3}$ -MMS allocation as long as no agent i values any item more than $\frac{\mu_i}{3}$. Thus, the real difficulty lies in distributing ‘high’ value items, i.e., items worth more than $\frac{\mu_i}{3}$ to some agent i . Drawing on the insights of [19, 12], we show a combination of maximum matching and greedy assignment suffices for this purpose. This gives our algorithm the basic structure: repeated maximum matching and greedy assignment to remove high valued items, followed by bag filling to allocate ‘low-value’ items. This approach allows for far simpler and more intuitive analysis than [6, 12, 31].

1.1.2 An Improved Algorithm

As the main result of this thesis, in Chapter 3, a new approach that gives a simple algorithm for showing the existence of a $\frac{3}{4}$ -MMS allocation is presented. Furthermore, the approach is powerful enough to be easily extended in two directions: First, we get a strongly polynomial time algorithm to find a $\frac{3}{4}$ -MMS allocation. Unlike previous work we do not need to use the PTAS in [48] to approximate the MMS values at all. Second, we show that there always exists a $(\frac{3}{4} + \frac{1}{12n})$ -MMS allocation, improving the best previous factor by Ghodsi et al. [31]. This improves the approximation guarantee, most notably for small n . We note that there are works, e.g., [6, 31], exploring better approximation factors for a small number of agents, and $\frac{3}{4}$ was the best factor known for $n > 4$.

The core of the algorithm is described in Section 3.2. It shows the existence of a $\frac{3}{4}$ -MMS allocation. We assume that MMS values are known for all agents. Since the MMS problem is scale-invariant (shown in Lemma 2.5), we scale valuations to make each agent’s MMS value 1. Then, we assign *high-value items* (e.g., a single item that some agent values at least $\frac{3}{4}$) to agents, who value them at least $\frac{3}{4}$, with a simple greedy approach based on the pigeonhole principle. We remove the assigned items and the agents receiving these items from further consideration. This reduces the number of high-value items to be at most $2n'$, where n' is the number of remaining agents. These greedy assignments massively simplify allocation of high-value items, which was the most challenging part of previous algorithms. Next, we prepare n' bags, one for each remaining agent, and put at most two high-value items in each bag. Then, we add low-value items on top of each of these bags one by one using a bag filling procedure until the value of bag for some agent is at least $\frac{3}{4}$. The main technical challenge here is to show that there are enough low-value items to give every agent a bag they value at least $\frac{3}{4}$.

In Section 3.3, the core algorithm is extended to compute a $\frac{3}{4}$ -MMS allocation in strongly polynomial time without any need to compute the actual MMS values. Here, a notion of *tentative assignments* is defined and a novel way for updating the MMS upper bound is presented. For each agent, the average value, that is the value of all items divided by the number of agents, is used as an upper bound of her MMS value. The only change from the basic algorithm is that some of the greedy assignments are tentative, i.e., they are valid only if the current upper bound of the MMS values is tight enough. It is shown in Section 3.3 that this can be checked by using the total valuation of low-value items. If the upper bounds are not tight enough for some agents, then we update the MMS value of such an agent and repeat. We show that we do not need to update the MMS upper bounds more than $O(n^3)$ times before we have a good upper bound on all MMS values. Then, we show that the same bag filling procedure, as in the basic algorithm, satisfy every remaining agent. The running time of the entire algorithm is $O(nm(n^4 + \log m))$.

In Section 3.4, It is shown that our basic algorithm also yields a better bound of the existence of a

$(\frac{3}{4} + \frac{1}{12n})$ -MMS allocation. The entire algorithm remains exactly the same but with an involved analysis. The analysis is tricky in this case, so a set of *dummy items* is added to make proofs easier. These items are used to make up for the extra loss for the remaining agents due to the additional factor, and, of course, these items are not assigned to any agent in the algorithm.

1.2 Groupwise Maximin Share (GMMS)

Besides MMS, we also studied a more general fairness notion called Group-wise Maximin Share (GMMS): for every subset of agents we want the sub-allocation also to be an MMS allocation (α -GMMS is defined accordingly). The study of GMMS allocations was initiated by [14] where they introduced an algorithm to obtain $\frac{1}{2}$ -GMMS allocations. In Chapter 4 the notion of Groupwise MMS (GMMS) is studied. The definition of GMMS allocation is presented in Chapter 4 (Definition 4.1).

In a GMMS allocation, the MMS guarantee is achieved not just with respect to M , but also among all the sub-groups of agents ($N' \subseteq N$) with respect to the items in the corresponding sub-allocation ($\bigcup_{i \in N'} A_i$). In other words, if $A = (A_1, \dots, A_n)$ is a GMMS (α -GMMS) allocation, then for every $N' \subseteq N$, the sub-allocation $(A_i)_{i \in N'}$ is an MMS (α -MMS) allocation.

We design an algorithm to obtain $\frac{4}{7}$ -GMMS allocations after donating at most n items that were intended to be allocated. This result was independently obtained by [24] at the same time and they kindly acknowledged our result in their work. Going beyond $\frac{4}{7}$ is one of the unanswered questions in this area.

1.3 MMS values in Mixed Manna

MMS problem is well-studied for the case of *good manna* (discussed above) where all items are valued non-negatively by every agent, and for the *bad (chore) manna* [11, 12, 33] where items are valued non-positively by everyone.

One natural extension to the *classic* good manna MMS problem is to assume that the valuations for items can be both negative and positive, i.e., the mixed manna setting. Many applications of fair division involve both goods that are freely disposable and chores that *have* to be assigned, for example, splitting assets and liabilities in partnership dissolution, and office tasks. We initiate the study of MMS allocation in the mixed manna settings [38]. This problem is more challenging than the good manna setting.

Note that in mixed manna setting, MMS value of the agents might be negative. In Chapter 5, MMS allocation is redefined to cover the case when the MMS value of the agents is negative. Below, we re-state the definition.

Definition 5.2 (α -MMS allocation). *A is called an α -MMS allocation, if for a given $\alpha > 0$, for each agent $i \in N$ we have $v_i(A_i) \geq \alpha\mu_i$ if $\mu_i \geq 0$, and $v_i(A_i) \geq (1/\alpha)\mu_i$, if $\mu_i < 0$. Equivalently,*

$$v_i(A_i) \geq \min\{\alpha\mu_i, (1/\alpha)\mu_i\}.$$

The problem of *finding the MMS value of an agent* is equivalent to the problem of *finding an MMS allocation for the case when the agents are identical*. The reason is that to obtain the MMS value of an agent, we need to obtain a partition based on the agent's valuation in a way that the value of the part with minimum value is maximized. Such partition will output n parts (bundles), where each bundle has a value of at least μ_i . Therefore, in the case of identical agents, an MMS partition is equivalent to MMS allocation.

Additionally, it implies that for the case of identical agents, MMS allocation (α -MMS allocation for $\alpha = 1$) always exists. In Chapters 5 and 6 it is assumed that agents are identical.

Finding MMS values of agents is NP-hard even in good manna setting but [48] gave a PTAS for this setting, i.e., an algorithm that finds $(1 - \epsilon)$ times the MMS values for any constant $\epsilon > 0$. Similarly, for chore manna [35] gave such a PTAS. In previous works, the PTAS to approximate MMS values has played an important role in algorithms for MMS problem with non-identical agents. A natural starting question is that is there a PTAS to find the MMS values of agents in a mixed manna setting? The rest of this thesis concentrates on identical agents in mixed manna.

In a separate result [38] (which is not discussed in this thesis), we proved that even in a highly restricted setting of two identical agents where the mixed manna has only two chores, it is NP-hard to find *approximate* MMS values within *any* constant factor. This was not a promising result. We further proved that in some settings, α -MMS allocation does not even exist for any $\alpha > 0$. The issue in mixed manna arises when the MMS values of the agents are too small compared to the absolute value of items. We overcome this challenge by restricting the problem setting either with extra assumptions to bound the MMS values or the value of items. Due to the hardness result, mixed manna can only be studied in restricted settings. While we have studied two directions, there are more to be discovered.

To describe these settings, we define two new notions. Let v^+ be total value of goods in M , i.e., $\sum_{j \in M: v_j \geq 0} v_j$ and v^- be total (absolute) value of chores in M i.e., $\sum_{j \in M: v_j < 0} |v_j|$.

For both settings, if the mentioned conditions hold, a PTAS to obtain MMS value of the agents is shown and if the conditions do not hold, then our algorithms correctly reports that.

Setting 1

In the first restricted setting which is studied in Chapter 5, we present a PTAS to approximate MMS values of the agents for any mixed MMS instance that satisfies two conditions. The two conditions are: (i) the number of agents n is a constant, and (ii) the total (absolute) value of agents for all the items (i.e., $|v(M)|$) is significantly greater than the minimum of their total value of goods (v^+) and their total (absolute) value for chores (v^-), i.e., for a constant $\tau > 0$, $|v_i(M)| \geq \tau \cdot \min\{v^+, v^-\}$. We show that the problem is tractable under these conditions, while [38] showed that dropping either renders the problem intractable.

thesis:mixed, first the sign of μ is identified and then the case of $\mu > 0$ and $\mu < 0$ are handled separately. We start by categorizing items into big and small based on their value. Then we show that if for an instance the aforementioned two conditions hold, then the number of big items is constant and, therefore, we can find all partitions of big items in constant time. Then, we show that only particular partitions are needed to find the MMS partition of identical agents. Next, we do a more generalized bag-filling on top of such partitions to add small items. Finally, we output the partition with highest value minimum bundle and we show that the value of minimum valued bundle is at least $\mu(1 - \epsilon)$ (or $\mu/(1 - \epsilon)$ when $\mu < 0$).

Setting 2

The reduction in [38] indicates that perhaps the bottleneck issue that makes the problem hard, is that the absolute MMS value can be arbitrarily small. Intuitively speaking, as the MMS value approaches arbitrarily close to zero, the problem of finding approximate MMS values approaches that of finding exact MMS values. In the limit where $\text{MMS} = 0$, every approximate MMS value is the exact MMS value. We then ask,

Question: If we had a guarantee that the absolute MMS value is greater than some threshold value, say Δ , then does the problem become tractable?

In the second restricted setting, we answer the above question positively when $|\mu| \geq \min\{v^+, v^-\}/\Delta$ for some constant $\Delta \geq 1$. We note that our assumption is weaker than having $\min\{v^+, v^-\}$ being a constant.

One of the key tools used by our algorithm is a carefully designed Integer Program (IP) that can be solved in polynomial-time. IPs have been used to solve related problems in several works. [48] gave a PTAS using this idea for the machine covering problem, which is equivalent to the MMS problem in goods manna. The MMS problem in chores manna (where all items are negatively valued by all agents) is equivalent to machine scheduling which has a PTAS using IP for bin packing (the dual problem). Several algorithms for the bin packing problem solve a relaxation of an IP as their main idea [25, 36, 37]. Our approach builds on these, but requires several new ideas to handle both goods and chores simultaneously.

1.4 Further Related Work

Classic Setting. Maximin share is a popular fairness notion of allocating indivisible items among agents. Bouveret and Lemaître [19] showed that an MMS allocation always exists in some restricted cases, e.g., when there are only two agents or if agents' valuations for items are either 0 or 1, but left the general case as an open problem. As mentioned earlier, Procaccia and Wang [45] showed that MMS allocation might not exist, but a $\frac{2}{3}$ -MMS allocation always exists. They also provided a polynomial time algorithm to find a $\frac{2}{3}$ -MMS allocation when the number of agents n is constant. For the special case of four agents, their algorithm finds a $\frac{3}{4}$ -MMS allocation. Amanatidis et al. [6] improved this result by addressing the requirement for a constant number of agents, obtaining a PTAS that finds a $(\frac{2}{3} - \epsilon)$ -MMS allocation for an arbitrary number of agents; see [40] for an alternate proof. In [6], they also showed that a $\frac{7}{8}$ -MMS allocation always exists when there are three agents. This factor was later improved to $\frac{8}{9}$ in [32].

Taking a different approach, Barman and Krishnamurthy [12] obtained a greedy algorithm to find a $\frac{2}{3}$ -MMS allocation. While their algorithm is fairly simple, the analysis is not.

Ghods et al. [31] improved these results by showing the existence of a $\frac{3}{4}$ -MMS allocation and a PTAS to find a $(\frac{3}{4} - \epsilon)$ MMS allocation.

Other variants of MMS. Maximin share fairness of good manna has also been studied in many different setting, e.g., for asymmetric agents (i.e., agents with different entitlements) [27], for group fairness [14, 24], beyond additive valuations [12, 31, 42], in matroids [32], with additional constraints [32, 16], for agents with externalities [22, 1], with graph constraints [15, 43], and with strategic agents [13, 5, 7, 10].

Chore Manna. With a chores manna, MMS values are negative, and an α -MMS allocation gives each agent i a bundle of value at least $\frac{1}{\alpha} \cdot \mu_i$. Haris Aziz et. al. [11] first studied the MMS problem with a chores manna. They introduced an algorithm for finding 2-MMS allocations (Our definition of α -MMS for the mixed manna is consistent for agents with positive as well as negative MMS values. We define α as smaller than 1, and consider $\frac{1}{\alpha}\mu$ valued bundles as α -MMS. Prior results for the chores manna have $\alpha > 1$ and ask for $\alpha \cdot \mu$ valued bundles. We state the approximation factors as defined in the original papers, and ask the reader to invert them when relating with ours). [12] improved the previous result by showing an algorithm for a $\frac{4}{3}$ -MMS allocation. Later, Barman and Krishnamurthy [33] improved this result to a $\frac{11}{9}$ -MMS allocation. They also showed a PTAS to find $(\frac{11}{9} + \epsilon)$ -MMS allocation and a polynomial time algorithm to find a $\frac{5}{4}$ -MMS allocation. In the chores only setting, weighted MMS [9], and asymmetric agents [8] notions have also been investigated.

Mixed Manna. Finding fair allocations of mixed items has recently caught a lot of attention for both divisible [17, 18] and indivisible [3, 4, 2, 29], items. However, to the best of our knowledge, ours is the first study on MMS allocations for a mixed manna.

Chapter 2

Preliminaries and A Simple Greedy Algorithm

2.1 Introduction

In this chapter, we formally define the MMS problem, we introduce some useful properties of the MMS problem, and, finally, we present an algorithm to find a $\frac{2}{3}$ -MMS allocation for agents with additive valuations. Our approach combines the insights of [19, 12] with the concepts developed in [31] to obtain an algorithm that is both simple to implement and analyze. Like [31], our algorithm consists of two phases: *matching* and *bag-filling*. However, unlike [31], our phases are much simpler, and we do not need to compute agents' MMS values. Bag-filling is a simple, greedy method to allocate 'low-value' items. We add one item at a time to a bag. If the value of the bag is worth at least $\frac{2}{3}$ of any agent's MMS, then we assign the bag to that agent, picking an arbitrary agent when there are more than one satisfying this condition. It is easily shown, in Section 2.3.1, that, in a normalized setting, bag-filling provides a $\frac{2}{3}$ -MMS allocation as long as no agent values any item more than $\frac{1}{3}$. Thus, the real difficulty lies in distributing 'high-value' items, i.e., items worth more than $\frac{1}{3}$ to some agent. Drawing on the insights of [19, 12], we show a combination of maximum matching and greedy assignment suffices for this purpose. This gives our algorithm the basic structure: repeated maximum matching and greedy assignment to remove high-value items, followed by bag-filling to allocate low-value items. Our approach allows for far simpler and more intuitive analysis than [6, 12, 31].

2.2 The MMS Problem

We consider the fair allocation of a set $M = \{1, \dots, m\}$ of m indivisible items among a set $N = \{1, \dots, n\}$ of n agents with additive valuations, using the popular notion of maximin share (MMS) as our measure of fairness. Let v_{ij} denote agent i 's value for item j , and i 's valuation of any bundle $S \subseteq M$ of items is given by $v_i(S) = \sum_{j \in S} v_{ij}$. Let $V = (v_1, \dots, v_n)$ denote the set of all valuation functions.

An agent's *MMS value* is defined as the maximum value she can guarantee herself if she is allowed to choose a partition of items into n bundles (one for each agent), on the condition that other agents choose their bundles from the partition before her. In the following definition we define it formally.

Definition 2.1 (MMS value and MMS partition). *Let $\mathcal{I} = \langle N, M, V \rangle$ denote an instance of the fair division*

problem, and let $\Pi_n(M) = \{P = \{P_1, \dots, P_n\} \mid P_i \cap P_j = \emptyset, \forall i, j; \cup_k P_k = M\}$ be the set of all feasible partitions of M into n bundles (one for each agent). Agent i 's MMS value or $\mu_i^n(M)$ (or simply μ_i when n and M are clear from the context) is defined as

$$\mu_i^n(M) = \max_{P \in \Pi_n(M)} \min_{P_k \in P} v_i(P_k) .$$

We call a partition achieving μ_i , an MMS partition of agent i .

Further, let $\mathcal{P}_i^n(M)$ denote the set of partitions achieving $\mu_i^n(M)$, i.e.,

$$\mathcal{P}_i^n(M) = \{P \in \Pi_n(M) : \min_{P_k \in P} v_i(P_k) = \mu_i^n(M)\} .$$

In other words, $\mathcal{P}_i^n(M)$ is set of all MMS partitions of agent i for items in M when there are n agents.

Definition 2.2 (α -MMS allocation and MMS problem). We say an allocation $A = (A_1, \dots, A_n)$ is α -MMS, for $\alpha \in (0, 1]$, if each agent i receives a bundle A_i worth at least α times her MMS value, i.e., $v_i(A_i) \geq \alpha \cdot \mu_i, \forall i \in N$. An MMS allocation is simply 1-MMS allocation.

Given an instance $\mathcal{I} = \langle N, M, V \rangle$ and an approximation factor $\alpha \in (0, 1]$, the MMS problem is to find an α -MMS allocation.

Now let's consider an example of an MMS problem instance with two agents and five items.

Example 2.3. Let $N = \{1, 2\}$ and $M = \{1, 2, 3, 4, 5\}$. The valuation of agents for items is shown in Table 2.1.

Table 2.1: Valuation of agent a for items in M in Example 2.3

Item $j =$	1	2	3	4	5
v_{1j}	3	1	2	5	4
v_{2j}	4	4	5	3	2

An MMS partition for agent 1 is: $P = \{P_1, P_2\}$ where $P_1 = \{1, 5\}$, $P_2 = \{2, 3, 4\}$. Since $v_1(P_1), v_1(P_2) \geq 7$ and we cannot find another partition with higher-value of lowest-valued bundle for agent 1, $\mu_1^2(M) = 7$.

An MMS partition for agent 2 is: $P' = \{P'_1, P'_2\}$ where $P'_1 = \{1, 3\}$, $P'_2 = \{2, 4, 5\}$. Since $v_2(P'_1), v_2(P'_2) \geq 9$ and we cannot find another partition with higher-value of lowest-valued bundle for agent 2, $\mu_2^2(M) = 9$.

An example of MMS allocation is $A = (A_1, A_2)$ where $A_1 = \{1, 4\}$ and $A_2 = \{2, 3, 5\}$. $v_1(A_1) = 8 \geq 7$ and $v_2(A_2) = 11 \geq 9$ so A is an MMS allocation.

2.3 Properties of Maximin Share

In this section, we state nice properties of maximin shares that our algorithm exploits. These are standard results appeared in [45, 6, 20, 12, 28, 31]. For the sake of completeness, we provide a brief proof of each property.

Lemma 2.4 (Average upper bounds MMS). $\mu_i^n(M) \leq \frac{v_i(M)}{n}, \forall i \in N$.

Proof. Suppose for some agent i , $\mu_i^n(M) > \frac{v_i(M)}{n}$, i.e., there exists a partition of M into n bundles where all bundles have value strictly more than $\frac{v_i(M)}{n}$. Therefore, $v_i(M) \geq n \cdot \mu_i^n(M) > n \cdot \frac{v_i(M)}{n} = v_i(M)$, a contradiction. $\square$

Lemma 2.5 (Scale Invariance). *Let $A = (A_1, \dots, A_n)$ be an α -MMS allocation for instance $\mathcal{I} = \langle N, M, V \rangle$. If we create an alternate instance $\mathcal{I}' = \langle N, M, V' \rangle$ where valuations of each agent i are scaled by $c_i > 0$, i.e., $v'_{ij} := c_i \cdot v_{ij}, \forall j \in M$, then $\mu'_i = c_i \cdot \mu_i$ and A is an α -MMS allocation for $\mathcal{I}'$.*

Proof. For any bundle $S \subseteq M$, we have $v'_i(S) = c_i \cdot v_i(S)$. Therefore, $\mu'_i = c_i \cdot \mu_i$. Further, $v'_i(A_k) = c_i \cdot v_i(A_k) \geq c_i \cdot \alpha \cdot \mu_i = \alpha \cdot \mu'_i, \forall k$. $\square$

We say agent i 's valuation is normalized for $\mathcal{I} = \langle N, M, V \rangle$ when her valuations are scaled so that $v_i(M) = n$ holds. In view of Lemma 2.5, normalizing agents' valuations provides a convenient upper bound on μ_i 's without affecting performance guarantees. In addition, this removes the problem of comparing the relative value of a bundle of items between agents whose scale of valuations differs in orders of magnitude.

Definition 2.6. *We call an instance $\mathcal{I} = \langle N, M, V \rangle$ normalized if $v_i(M) = n$ for all $i \in N$.*

In the next two following sections, we build intuition for what exactly makes finding α -MMS allocations difficult.

2.3.1 Bag-filling for Low-value Items

Suppose we seek an $(1 - \delta)$ -MMS allocation. If we normalize the valuation of agents so that $v_i(M) = n, \forall i \in N$ and we have $v_{ij} \leq \delta, \forall i \in N, j \in M$, then we can find a $(1 - \delta)$ -MMS allocation using the following simple *bag-filling procedure*: All items are unallocated in the beginning. Start with an empty bag B and keep filling it with unallocated items until some agent a values B at least $(1 - \delta)$. Then, allocate B to a (choose a arbitrarily if there are multiple such agents). We repeat this process for the set of unallocated items and the set of agents who have not allocated any bundle yet. In Proposition 1 we show that this procedure will give every agent a bundle worth at least $\frac{2}{3}\mu_i$. We note that a similar result also appears in [31].

Proposition 1. *Assume agents' valuations are normalized as defined in (2.6), and that no agent values any item more than $0 < \delta < \frac{1}{2}$: $v_{ij} \leq \delta$ for all $j \in M$, for all $i \in N$. Then, the bag-filling algorithm gives a $(1 - \delta)$ -MMS allocation.*

Proof. By the definition of normalized valuations, we have $v_i(M) = n$ and $\mu_i \leq 1$ for all $i \in N$. Therefore, it is enough to show each agent receives a bag worth at least $1 - \delta$. Clearly, the agent that receives the bag in each iteration gets at least $1 - \delta$, so the claim amounts to showing remaining agents do not lose too much value when the bag is assigned. Let j be the last item added to the bag B . Note that before adding j , all agents valued B less than $1 - \delta$, i.e., $v_i(B \setminus \{j\}) < 1 - \delta$ for all $i \in N$. Now, since valuations are additive and $v_{ij} < \delta$ for all agents $i \in N$, we have $v_i(B) \leq 1$. That is, no agent values the bag B more than 1. This means after removing agent a and the items of B , all remaining agents satisfy $v_i(M \setminus S) = v_i(M) - v_i(B) \geq n - 1$. Since initially $v_i(M) = n, \forall i$, there are enough items to satisfy all the agents with a bag that they value $(1 - \delta)$. $\square$

We can conclude from Proposition 1 that if the agents' valuations is normalized and all items value less than $\frac{1}{3}$ ('low-value') for all agents, then it is straightforward to find a $\frac{2}{3}$ -MMS allocation.

Later in this chapter and also in Chapter 3, we design more general bag-filling procedures.

2.3.2 Reduction

A useful concept of *valid reduction* is used in [39, 40, 6, 31, 28]. From Definition 2.1, $\mu_i^k(S)$ denote the MMS value of agent i when S is the set of items that needs to be divided among k agents (including i). Recall that for the α -MMS allocation problem for instance $\mathcal{I} = \langle N, M, V \rangle$, we want to partition M into n bundles $(A_1, \dots, A_n)$ such that $v_i(A_i) \geq \alpha \cdot \mu_i^n(M), \forall i$.

Definition 2.7 (Valid reduction). *For obtaining an α -MMS allocation, the act of removing a set $S \subseteq M$ of items and an agent i from M and N is called a valid reduction if*

$$\begin{aligned} v_i(S) &\geq \alpha \cdot \mu_i^n(M) \\ \mu_{i'}^{n-1}(M \setminus S) &\geq \mu_{i'}^n(M), \quad \forall i' \in N \setminus \{i\}. \end{aligned} \quad (2.1)$$

In words, valid reduction is the process of reducing the size of the instance $\mathcal{I} = \langle N, M, V \rangle$ by assigning a set of items S to an agent i and getting a smaller instance $\mathcal{I}' = \langle N \setminus \{i\}, M \setminus S, V \setminus \{v_i\} \rangle$ while the two conditions in (2.1) is satisfied. Clearly, an α -MMS allocation for the smaller instance $\mathcal{I}'$ gives an α -MMS allocation for the original instance. In our algorithms, we use it to remove *high-value items* and get smaller instances.

Proposition 2. *Let $\mathcal{I} = \langle N, M, V \rangle$ be a problem instance, and let μ_i be the MMS value for agent $i \in N$. If we remove one agent $a \in N$ and one item $j \in M$ where $v_{aj} \geq \frac{2}{3}$, then the MMS value of all remaining agents in the reduced instance $\mathcal{I}' = \langle N \setminus \{a\}, M \setminus \{j\}, V \rangle$, is at least as large as their MMS value in $\mathcal{I}$ i.e., $\mu'_i \geq \mu_i$. In words, removing one agent and one item from a problem instance this is a valid reduction.*

Proof. Suppose agent $a \in N$ and item $j \in M$ are removed from the instance, and let i be any remaining agent. Consider an MMS partition $P = \{P_1, \dots, P_n\}$ of agents i that she makes to calculate her MMS value in the original instance $\mathcal{I}$, and note that $v_a(P_k) \geq \mu_i$ for all $P_k \in P$ by the definition of MMS value. In the reduced instance $\mathcal{I}' = \langle N \setminus \{a\}, M \setminus \{j\}, V \rangle$, agent i needs to make one less bundle but has one less item. Let $P_{k^*} \in P$ be the bundle containing the removed item j . Suppose she simply takes the items of $P_{k^*} \setminus \{j\}$ and distributes them arbitrarily to the other bundles of P to create a new allocation $\hat{P} = \{\hat{P}_1, \dots, \hat{P}_{n-1}\}$. Clearly, $\hat{P}$ is a feasible allocation, and $\hat{P}_k \geq \mu_i$ for all $\hat{P}_k \in \hat{P}$. Therefore, $\mu'_i \geq \mu_i$. $\square$

Proposition 2 shows that removing one agent and one item does not reduce MMS guarantees for remaining agents in the reduced instance. It is straightforward to generalize the above argument to show that removing k agents and k items does not decrease MMS guarantees in the reduced instance.

Corollary 2.8. *Consider and MMS instance $\mathcal{I} = \langle N, M, V \rangle$. Allocating k items to k agents (one item for each agent) where agents like their own allocated item at least $\frac{2}{3}$ and removing such items and agents from the instance $\mathcal{I}$ is a valid reduction.*

Combining Propositions 1 and 2 yields a simple greedy algorithm to compute a $\frac{1}{2}$ -MMS allocation. We start by normalizing valuations as defined in Definition 2.6. By Lemma 2.4, this ensures $\mu_i \leq 1$ for all agents $i \in N$. If some agent, say a , has valuation $v_{aj} \geq \frac{1}{2}$ for some item j , then we assign item j to agent a . If more than one agent satisfies this condition, then pick one arbitrarily. By Proposition 2, this is a valid reduction. Next, we normalize valuations for the reduced instance $\mathcal{I}' = \langle N \setminus \{a\}, M \setminus \{j\}, V \rangle$, and repeat the process, greedily assigning one item at a time to any agent who values the item at least $\frac{1}{2}$ and then normalizing valuations, until either all agents are removed or $v_{ij} < \frac{1}{2}, \forall j \in M, \forall i \in N$. In the later case, Proposition 1

shows that the bag-filling algorithm provides all remaining agents a bundle worth at least $\frac{1}{2}$ of their maximin share.

A natural approach to extending the above algorithm to give a $\frac{2}{3}$ -MMS allocation requires splitting the items M into three sets based on their value: high-value items for which $v_{ij} \geq \frac{2}{3}$ for some agent $i \in N$, low-value items for which $v_{ij} < \frac{1}{3}$ for all agents $i \in N$, and medium-value items for which $v_{ij} \geq \frac{1}{3}$ for at least one agent $i \in N$ but $v_{ij} < \frac{2}{3}$ for all $i \in N$. Notice that, similar to the $\frac{1}{2}$ -MMS algorithm above, we may greedily assign high-value items to give at least $\frac{2}{3}\mu_i$ to the agent receiving the item without decreasing the MMS of any remaining agent in the reduced instance (valid reduction). Also, if all items are low-value $v_{ij} < \frac{1}{3} \forall i \in N$, then the bag-filling algorithm easily yields a $\frac{2}{3}$ -MMS allocation. The real challenge lies in managing the medium-value items. These items are not valuable enough individually to satisfy $\frac{2}{3}\mu_i$ for an agent i , yet they are too valuable, to some agent, to distribute haphazardly through bag-filling. Thus, we seek a simple, efficient means to allocate medium-value items.

In the next section we introduce a property of MMS problem which helps us to manage medium-value items.

2.3.3 Ordered Instances

We say that an instance $\mathcal{I} = \langle N, M, V \rangle$ is ordered if:

$$v_{i1} \geq v_{i2} \geq \dots \geq v_{im}, \quad \forall i \in N. \quad (2.2)$$

In words, in an ordered instance, all agents have the same order of preferences over items. Roughly speaking, this maximizes the competition between agents, and, intuitively, should make it more difficult to provide an MMS allocation. Indeed, Bouveret and Lemaître [20] show ordered instances are worst case. They provided a reduction from any arbitrary instance $\mathcal{I} = \langle N, M, V \rangle$ to an ordered instance $\mathcal{I}' = \langle N, M, V' \rangle$ with a simple polynomial-time procedure that converts an MMS allocation of $\mathcal{I}'$ into an MMS allocation of $\mathcal{I}$. Later, Barman and Krishnamurthy [12] generalized this result for α -MMS allocations. This property is used in [12, 28] to find a $\frac{2}{3}$ -MMS allocation. Observe that the MMS values of an agent in $\mathcal{I}$ and $\mathcal{I}'$ are the same because, by Definition 2.1, it neither depends on the order of the items nor on other agents' valuations.

Proposition 3 (Section 2.1 of Barman and Murthy [12]). *Given any instance $\mathcal{I} = \langle N, M, V \rangle$, one can find its corresponding ordered instance $\mathcal{I}' = \langle N, M, V' \rangle$ in polynomial time.*

Algorithm 1 gives explicit details for the process of converting any instance $\mathcal{I}$ into an ordered instance $\mathcal{I}'$. We call $\mathcal{I}'$ constructed this way an ordered instance.

Algorithm 1: Converting to an Ordered Instance

Input : Original Instance $\mathcal{I} = \langle N, M, V \rangle$

Output : V' : Valuations for Ordered Instance

```

1 for  $j = 1$  to  $m$  do
2   for  $i = 1$  to  $n$  do
3      $j^* =$  agent  $i$ 's  $j$ th most valuable item ;
4      $v'_{ij} \leftarrow v_{ij^*}$  ;
```

Theorem 2.9 (Theorem 2 and Corollary 1 of Barman and Murthy [12]). *For any instance $\mathcal{I}$, let $\mathcal{I}'$ be its ordered instance. If A' is an α -MMS allocation for $\mathcal{I}'$, then using A' we can find allocation A which is an α -MMS allocation for $\mathcal{I}$ in polynomial time.*

Algorithm 2 shows how to obtain an α -MMS allocation A for the original instance $\mathcal{I}$ given an α -MMS allocation for the ordered instance $\mathcal{I}'$. For the sake of completeness, we provide a brief proof of Theorem 2.9.

Algorithm 2: α -MMS Allocation for Unordered Instance

Input : Allocation $A' = (A'_1, \dots, A'_n)$ for Ordered Normalized Instance $I' = (N, M, V')$ such that $v'_i(A'_i) \geq \alpha$ for all $i \in N$.

Output: Allocation $A = (A_1, \dots, A_n)$ for Original Normalized Instance $I = (N, M, V)$ such that $v_i(A_i) \geq \alpha$ for all $i \in N$.

```

1  $A = (\emptyset, \dots, \emptyset)$  and  $R \leftarrow M$  ;
2 for  $j = 1$  to  $m$  do
3    $a \leftarrow i : j \in A'_i$  (pick the agent assigned item  $j$  in  $A'$ ) ;
4    $g \leftarrow \arg \max_{k \in R} v_{ak}$  ;
5    $A_i \leftarrow A_i \cup \{g\}$  and  $R \leftarrow M \setminus \{g\}$  ;
```

Proof. (Theorem 2.9) Clearly, both Algorithms 1 and 2 run in polynomial time. Notice that Algorithm 2 allocates each item $j \in M$ to at most one agent $i \in N$ and that one item is allocated in each iteration. Let k_j be the item allocated in the j th iteration of Algorithm 2, lines 2 through 5. Consider the agent i assigned $j \in A'_i$, meaning that $k_j \in A_i$. At the beginning of the j th iteration, exactly $j-1$ items have been allocated. Therefore, k_j is among the top j most valuable items for agent i . From the construction of the ordered instance I' , it follows that for all $j \in A'_i$, $v_i(k_j) \geq v'_i(j)$. Therefore, $v_i(A_i) = \sum_{j \in A'_i} v_i(k_j) \geq \sum_{j \in A'_i} v'_i(j) = v'_i(A'_i) \geq \alpha$. $\square$

Lemma 2.10. (ORDERED INSTANCE [20, 12]). *Without loss of generality, we can assume that agents have the same order of preferences over the items, i.e., $v_{i1} \geq v_{i2} \geq \dots \geq v_{im}, \forall i \in N$.*

Lemma 2.10 shows that it suffices to consider ordered instances. A total ordering over the set of items M provides precious information to us as algorithm designers since we know precisely which items are best (favored by all agents). In other words, the ordering over M essentially means all items fall into three categories: low, medium, and high-value, corresponding to low, medium, and high in the ordering respectively. In Section 2.4, we show that a total ordering over the items M allows for a simple generalization of the $\frac{1}{2}$ -MMS allocation algorithm described in Section 2.3.2 to give $\frac{2}{3}$ -MMS allocations.

2.4 A 2/3-MMS Approximation

In this section, we present an algorithm to find $\frac{2}{3}$ -MMS allocations. Our method involves a preprocessing step with Proposition 3 to ensure the instance is ordered. We show how to obtain a $\frac{2}{3}$ -MMS allocation for the ordered instance, then use a post processing step with Theorem 2.9 to obtain a $\frac{2}{3}$ -MMS allocation for the original instance. From this point on, we assume the instance is ordered as defined in (2.2).

The algorithm builds one bundle of items at a time, assigns it to some agent i who values it at least $\frac{2}{3}\mu_i$, and then removes that agent and the bundle from the instance. The basic structure of the algorithm closely resembles the simple $\frac{1}{2}$ -MMS algorithm discussed in Section 2.3.2. In fact, the same simple strategies guide the algorithm's design.

Assuming valuations are normalized as defined in Definition 2.6, our algorithm handles allocation of items based on their value: low, medium or high. For this we use the clustering approach of [31], and define the following sets of items:

$$\begin{aligned} J_H &= \{j \in M : \exists i \in N \text{ s.t. } v_{ij} \geq \frac{2}{3}\} \\ J_M &= \{j \in M : \exists i \in N \text{ s.t. } \frac{1}{3} \leq v_{ij}, v_{ij} < \frac{2}{3}, \forall i \in N\} \\ J_L &= \{j \in M : v_{ij} < \frac{1}{3}, \forall i \in N\}, \end{aligned} \tag{2.3}$$

which correspond to high, medium, and low-value items respectively. Second, for any bundle $S \subseteq M$, we define the set $N(S)$ as the agent's with value at least $\frac{2}{3}$ for S :

$$N(S) = \{i : i \in N, v_i(S) \geq \frac{2}{3}\}. \tag{2.4}$$

By using the preprocessing step of Proposition 3, we ensure a total ordering on the items (2.2). Thus, for any agent i if $v_{ij} > \frac{1}{3}$ for some item j , then $v_{ij'} > \frac{1}{3}$ for all $j' \leq j$. Similarly, if $v_{ij} < \frac{2}{3}$, then $v_{ij'} < \frac{2}{3}$ for all $j' \geq j$.

2.4.1 2/3-MMS Algorithm

At a high level, our algorithm mirrors the simple $\frac{1}{2}$ -MMS algorithm, consisting of two phases: matching and bag-filling. Like the $\frac{1}{2}$ -MMS algorithm, we allocate high-value items J_H through a maximum matching, and assign all low-value items J_L through bag-filling. For medium-value items J_M , the total ordering on the items simplifies allocation decisions based on $|J_M|$. If $|J_M|$ is sufficiently large, we greedily assign a bundle containing the two 'least valuable' items of $|J_M|$ to any agent that values it at least $\frac{2}{3}$, using a generalization of Proposition 2. Otherwise, we use a modified version of the bag-filling algorithm. We make the treatment of medium-value items more precise shortly, but note that, the total ordering of items allows for small adjustments to the matching and bag-filling stages of the $\frac{1}{2}$ -MMS algorithm to improve the approximation guarantees to $\frac{2}{3}$ -MMS. Further, our approach makes the analysis of each stage nearly as simple as in the $\frac{1}{2}$ -MMS algorithm. We now explain the algorithm in more detail, see Algorithm 3 for a formal description.

Matching Procedure. The initial phase of the algorithm allocates high-value items J_H through a maximum matching we call the Matching Procedure. First, we normalize valuations which ensures $\mu_i \leq 1$ for all agents by Lemma 2.4. Next, we form a bipartite graph with agents on the left hand side and items of J_H on the right. We create an edge between agent i and item j , if $v_{ij} \geq \frac{2}{3}$. In words, the graph's edges connect agents with items they value at least $\frac{2}{3}$. Next, we solve a maximum matching T , and assign i bundle $A_i = \{j\}$, if $(i, j) \in T$. All matched items and agents are removed from the instance and we normalize valuations for the remaining agents. This process repeats until $|J_H| = 0$, i.e., there are no more high-value items.

Greedy Assignment from J_M . After completing the first phase, all high-value items are allocated. Next, we determine how to distribute medium-value and low-value items among the remaining agents. Our preprocessing step with Proposition 3 ensures the instance is ordered (2.2), meaning there is a least preferred item in any set of items (2.3). More precisely, $j^* = \arg \max_{j \in J_M} j$ is the least preferred item of J_M (medium-value items). When $|J_M| > n$, each agent must create at least one bundle containing two or more items of J_M when calculating their μ_i , by pigeon hole principle. Similar to the matching stage of the $\frac{1}{2}$ -MMS algorithm, we greedily assign the two least preferred items of J_M , $S = \{j^* - 1, j^*\}$, to an arbitrary

agent a with valuation $v_a(S) \geq \frac{2}{3}$. This ensures the a receiving S gets at least $\frac{2}{3}\mu_a$, and the MMS of all remaining agents i in the reduced instance $\mathcal{I}' = \langle N \setminus \{a\}, M \setminus S, V \rangle$ satisfy: $\mu'_i \geq \mu_i$. Agent's valuations are then normalized, and process repeats.

Modified Bag-filling. After allocating the bulk of medium-value items $|J_M| \leq n$, we create bundles for the remaining agents through a slightly modified version of bag-filling. Here, we simply initialize the bag S using one, arbitrary item from J_M , and then fill the bag with items from J_L (low-value items) until some agent i values S at least $v_i(S) \geq \frac{2}{3}$. Once $|J_M| = 0$, we use the standard bag-filling algorithm.

Recall that the challenge of improving the approximation guarantees of the $\frac{1}{2}$ -MMS algorithm requires proper management of the medium-value items J_M . Our approach, using Proposition 3 and Theorem 2.9 to ensure the instance is ordered, enables a simple and natural extension of the straight-forward $\frac{1}{2}$ -MMS algorithm to provide improved $\frac{2}{3}$ -MMS guarantees.

Algorithm 3: $\frac{2}{3}$ -MMS Allocation

Input : Ordered Instance $\langle N, M, V \rangle$
Output : $\frac{2}{3}$ -MMS Allocation

```

1 while  $|J_H| > 0$  or  $|J_M| > n$  do
2   Normalize Valuations ;
3   if  $|J_H| > 0$  then
4     Matching Procedure ;
5   else
6      $j^* \leftarrow \max_{j \in J_M} j$ ; // lowest value item of  $J_M$ 
7      $N(j^*) \leftarrow \{i : i \in N, v_i(j^*, j^* - 1) \geq \frac{2}{3}\}$  ;
8      $i \in N(j^*)$ ;  $A_i \leftarrow \{j^*, j^* - 1\}$ ; // assign  $i$  the bundle  $\{j^*, j^* - 1\}$ 
9      $N \leftarrow N \setminus \{i\}$ ;  $M \leftarrow M \setminus \{j^*, j^* - 1\}$  ;
10  while  $n \geq 0$  do
11    if  $|J_M| > 0$  then
12       $S \leftarrow j \in J_M$ ; // create a bag with arbitrary item of  $J_M$ 
13    else
14       $S \leftarrow \emptyset$ ; // create an empty bag
15       $N(S) = \{i : i \in N, v_i(S) \geq \frac{2}{3}\}$ ; //  $N(S)$  changes with  $S$ 
16      while  $|N(S)| = 0$  do
17         $j \in J_L$ ;  $S \leftarrow S \cup j$ ; // add arbitrary low-value item to the bag
18         $i \in N(S)$ ;  $A_i \leftarrow S$ ; // assign  $i$  the bundle  $S$ 
19         $N \leftarrow N \setminus \{i\}$ ;  $M \leftarrow M \setminus S$  ;

```

The algorithm consists of two phases, matching and bag-filling. The phases use different allocation procedures based on $|J_H|$ and $|J_M|$ respectively. We consider these procedures separately, starting with the matching procedure.

Lemma 2.11. *Let $\mathcal{I} = \langle N, M, V \rangle$ be a problem instance where agents' valuations are normalized and let μ_i be agent i 's MMS value. Suppose $|J_H| > 0$, as defined in (2.3), and that the Matching Procedure is used to create a maximum matching T . Let L be the agents of T and S be the items of T . Then,*

i) $|L| = |S| > 0$.

ii) All removed agents i receive at least $\frac{2}{3}\mu_i$.

ii) Assigning the items to the agents that they are matched to in T and removing them from $\mathcal{I}$, is valid reduction.

Proof. Recall the Matching Procedure creates a bipartite graph $G = (V, E)$ where the vertices V consist of agents on the left side and items on the right. An edge $e \in E$ is created between agent i and item j if $v_{ij} \geq \frac{2}{3}$. Finally, a maximum matching T is determined. By definition of J_H and the fact $|J_H| > 0$, the set of edges E of G is non-empty. Since T is a maximum matching, part i) is obvious. Next, recall that Lemma 2.4 shows that $\mu_i \leq 1$ for all $i \in N$ since valuations are normalized. Part ii) then follows by the construction of G . Finally, since $|L| = |S|$, Corollary 2.8 guarantees allocating the matched items is a valid reduction. $\square$

We now consider the second procedure of the algorithm's matching phase.

Lemma 2.12. *Let $\mathcal{I} = \langle N, M, V \rangle$ be an ordered problem instance with normalized valuations, and let μ_i be agent i 's MMS value. Suppose that $|J_H| = 0$ and $|J_M| > n$. Define $j^* = \arg \max_{j \in J_M} j$, and let $S = \{j^*, j^* - 1\}$ be the set of two least preferred items of J_M . Suppose bundle S is assigned to an arbitrary agent a satisfying $v_a(S) \geq \frac{2}{3}$. Then,*

i) $v_a(S) \geq \frac{2}{3}\mu_a$.

ii) Removing j^* and $j^* - 1$ from M and a from N is a valid reduction.

Proof. The argument is a simple generalization of Proposition 2. From normalized valuations and Lemma 2.4, $\mu_i \leq 1$ for all $i \in N$. By definition of the set J_M , for all items $j \in J_M$ there exists an agent $a \in N$ so that $v_{aj} \geq \frac{1}{3}$. Since the instance is ordered, if $v_{aj} \geq \frac{1}{3}$, then $v_{aj'} \geq \frac{1}{3}$ for all $j' \leq j$. Since $|J_M| > n > 0$ and $j^* \in J_M$, there exists at least one agent $a \in N$ so that $v_a(S) \geq \frac{2}{3}\mu_a$, showing part i).

We now show part ii). For any remaining agent i in the reduced instance $\mathcal{I}'$, consider an MMS partition $P = \{P_1, \dots, P_n\}$ she makes while computing her μ_i in the original instance $\mathcal{I}$. Note that $v_i(P_k) \geq \mu_i$ for all $P_j \in P$. In the reduced instance $\mathcal{I}'$, agent i must create one less bundle, but has two fewer items, specifically j^* and $j^* - 1$. We show how to construct a feasible partition $A' = \{P'_1, \dots, P'_{n-1}\}$ so that $v_i(P'_j) \geq \mu_i$ for all $P'_j \in P'$. Notice that the condition $|J_M| > n$ guarantees that at least one bundle, say P_k , must contain at least two items, say $u, v \in J_M$, by the pigeon hole principle. Without loss of generality, we may assume $v_{iu} \leq v_{iv}$. Since we take the two lowest valued items of J_M , $S = \{j^*, j^* - 1\}$, then $v_{ij^*} \leq v_{iu}$ and $v_{ij^*-1} \leq v_{iv}$. Let P_{j^*} and P_{j^*-1} be the bundles of P containing items j^* and $j^* - 1$ respectively. Suppose agent i swaps item $u \in P_k$ with item $j^* \in P_{j^*}$ and swaps item $v \in P_k$ with item $j^* - 1 \in P_{j^*-1}$ to create P'_k , P'_{j^*} , and P'_{j^*-1} . Finally, i distributes the items of $P'_k \setminus S$ to other bundles arbitrarily to create a new set of bundles $P' = \{P'_1, \dots, P'_{n-1}\}$. It is clear that P' is a feasible allocation and that $v_i(P'_k) \geq \mu_i$. Therefore, agent i 's MMS value μ'_i in the reduced instance $\mathcal{I}'$ satisfies $\mu'_i \geq \mu_i$. Therefore, removing S from M and a from N is a valid reduction $\square$

We now consider the algorithm's second phase, bag-filling.

Lemma 2.13. *Let $\mathcal{I} = \langle N, M, V \rangle$ be an ordered problem instance with normalized valuations. Suppose that $|J_H| = 0$ and $0 < |J_M| \leq n$. Then, the modified bag-filling algorithm ensures all agents receive a bundle worth at least $\frac{2}{3}$ of their MMS value.*

Proof. This argument is a simple generalization of Proposition 1. In modified bag-filling, we simply initialize the bag S to an arbitrary item $j \in J_M$. Notice that, this initialization ensures the condition $|J_M| \leq n$ holds

in each iteration since we always remove one agent and one item of J_M . As valuations are normalized, it is enough to show all agents receive a bundle worth at least $\frac{2}{3}$.

First, note that $|J_L| > 0$, since from normalized valuations and the fact that $v_{ij} < \frac{2}{3} \forall j \in J_M$, we see that $\forall i \in N$: $v_i(J_L) = v_i(M) - v_i(J_M) \geq n - \frac{2}{3}|J_M| \geq \frac{n}{3} > 0$. We now show that some agent i eventually values the bag $v_i(S) \geq \frac{2}{3}$. Let $j \in J_M$ be the initial item of the bag. If there exists an agent $i \in N$ such that $v_{ik} < \frac{1}{3}$ for all $k \in M$, then, clearly there exists some $S' \subset J_L$ so that $v_i(j \cup S') \geq \frac{2}{3}$. Suppose that no such agent exists. Note that from the definition of J_M , there exists some agent i such that $v_{ij} \geq \frac{1}{3}$. Given that $v_i(J_L) \geq \frac{n}{3}$, we see that $v_i(j \cup J_L) \geq \frac{1}{3} + \frac{n}{3} \geq \frac{2}{3}$ for $n \geq 1$. Therefore, there exists $S' \subset J_L$ so that $v_i(j \cup S') \geq \frac{2}{3}$. This establishes the bag is eventually assigned to an agent who values it at least $\frac{2}{3}$.

Let a be the agent assigned the bag S . Now, we show that $v_i(S) \leq 1$ for all other agents $i \in N \setminus \{a\}$. Before adding the final item of the bag $j' \in S$, $v_i(S \setminus j') < \frac{2}{3}$ for all $i \in N$. The final item added to the bag comes from J_L so $v_{ij'} < \frac{1}{3}$ for all $i \in N$. Therefore, $v_i(S) < 1$ for all $i \in N$. This means that for each agent i , $v_i(M) \geq n$ and $v_i(J_L) \geq \frac{n}{3}$ are invariants of the algorithm. Then, it is easy to see all agents receive a bundle worth $\frac{2}{3}$. Finally, when $|J_M| = 0$, all agents receive a bundle worth at least $\frac{2}{3}$ by Proposition 1. $\square$

From Lemmas 2.11, 2.12, and 2.13, we get the following theorem.

Theorem 2.14. *Algorithm 3 provides a $\frac{2}{3}$ -MMS allocation.*

Remark 2.15. *Lemmas 2.12 and 2.13 are really just simple generalizations of Propositions 2 and 1 (respectively) designed to manage medium-value items J_M . In this sense, our algorithm is natural generalization of the simple $\frac{1}{2}$ -MMS algorithm of Section 2.3.2 which improves performance guarantees to $\frac{2}{3}$ MMS.*

2.5 Discussion

In this chapter we investigate fair division of indivisible items using maximin share as our measure of fairness of an allocation. We propose a simple greedy approximation algorithm to obtain a $\frac{2}{3}$ -MMS allocation. Further, we show that our algorithm can be seen as a natural extension of the $\frac{1}{2}$ -MMS algorithm discussed in Section 2.3.2. This allows for a far simpler, and more intuitive analysis as compared to other existing $\frac{2}{3}$ -MMS approximations.

Generalizing this approach to to provide better performance guarantees does not seem straightforward. Consider designing an algorithm to give a $\frac{3}{4}$ -MMS allocation. Suppose we naively create three clusters of items: high $v_{ij} \geq \frac{3}{4}$ for at least one agent i , medium $v_{ij} \geq \frac{1}{4}$ for at least one agent but $v_{ij} < \frac{3}{4}$ for all agents, and low $v_{ij} < \frac{1}{4}$ for all agents. Similar to the $\frac{2}{3}$ case, we allocate high-value items through maximum matching, and if all items are low-value, then bag-filling suffices to distribute all remaining items. Notice that, we must assign two or three medium-value items to ensure an agent receives at bundle worth at least $\frac{3}{4}$. If $|J_M| > 2n$, then we can guarantee each agent must create at least one bundle containing three items from J_M when computing their MMS, and therefore, may justify greedily assigning a bundle containing the three lowest valued items of J_M to any agent who values it at least $\frac{3}{4}$. When $2n \geq |J_M| > n$, the situation is less clear. We know each agent creates at least one bundle containing two items from J_M when computing their MMS, but we can't guarantee that some agent will value a bundle containing only the two lowest valued items of J_M at least $\frac{3}{4}$. Further, we can't guarantee that we may start bag-filling where we initialize the bag to the two least valuable items of J_M since some agent might value some set of two 'better' (more valuable) items of J_M more than 1. If we initialize the bag to only the lowest valued item of J_M , then we might 'run'

out of low-value items, leaving only medium-value items and no way to ensure each remaining agent receives at least $\frac{3}{4}$.

Attempting a finer partitioning of J_M significantly complicates analysis as it creates numerous special cases based on the number of items within each sub-cluster of medium-value items. Further, it is not clear that a simple allocation decision exists for all possible special cases. For these reasons, it seems the approach presented in this chapter is only capable of producing a $\frac{2}{3}$ -MMS allocation. However, as our algorithm is closely related to the simple $\frac{1}{2}$ -MMS approximation, we find our approach more intuitive than other existing $\frac{2}{3}$ -MMS algorithms.

In the next chapter, we will show how to overcome these difficulties with the help of new ideas and more involved analysis to obtain a better guarantee for MMS approximation.

Chapter 3

A $3/4$ -MMS Algorithm

3.1 Introduction

In this chapter, we use properties that were introduced in Chapter 2 and the intuition built while obtaining $\frac{2}{3}$ -MMS algorithm in the aforementioned chapter, and we develop a new approach that gives a simple algorithm for showing the existence of a $\frac{3}{4}$ -MMS allocation. Furthermore, our approach is powerful enough to be easily extended in two directions: First, we get a strongly polynomial time algorithm to find a $\frac{3}{4}$ -MMS allocation, where we do not need to use the PTAS in [48] to approximate the MMS values at all. Second, we show that there always exists a $(\frac{3}{4} + \frac{1}{12n})$ -MMS allocation, improving the best previous factor by Ghodsi et al. [31]. This improves the approximation guarantee, most notably for small n . We note that there are works, e.g., [6, 31], exploring better approximation factors for a small number of agents, and $\frac{3}{4}$ was the best factor known for $n > 4$.

Our algorithms are extremely simple. We first describe the basic algorithm, given in Section 3.2, that shows the existence of a $\frac{3}{4}$ -MMS allocation. We assume that MMS values are known for all agents. Since the MMS problem is scale-invariant (shown in Lemma 2.5), we scale valuations to make each agent's MMS value 1. Then, similar to what we saw in Chapter 2, we assign *high-value items* (e.g., a single item that some agent values at least $\frac{3}{4}$) to agents, who value them at least $\frac{3}{4}$, with a simple greedy approach based on the pigeonhole principle. We remove the assigned items and the agents receiving these items from further consideration. This reduces the number of high-value items to be at most $2n'$, where n' is the number of remaining agents. These greedy assignments massively simplify allocation of high-value items, which was the most challenging part of previous algorithms. Next, we prepare n' bags, one for each remaining agent, and put at most two high-value items in each bag. Then, we add low-value items on top of each of these bags one by one using a bag-filling procedure until the value of bag for some agent is at least $\frac{3}{4}$. The main technical challenge here is to show that there are enough low-value items to give every agent a bag they value at least $\frac{3}{4}$.

In Section 3.3, we extend the basic algorithm to compute a $\frac{3}{4}$ -MMS allocation in strongly polynomial time without any need to compute the actual MMS values (using the PTAS in [48]). Here, we define a notion of *tentative assignments* and a novel way for updating the MMS upper bound. For each agent, we use the average value, that is the value of all items divided by the number of agents, as an upper bound of her MMS value. The only change from the basic algorithm is that some of the greedy assignments are tentative, i.e., they are valid only if the current upper bound of the MMS values is tight enough. We show that this can

Algorithm 4: α -MMS Allocation

Input : Ordered Instance $\mathcal{I} = \langle N, M, V \rangle$, i.e., $v_{i1} \geq v_{i2} \geq \dots \geq v_{im}, \forall i \in N$ and α

Output : α -MMS Allocation

- | | |
|---|---|
| 1 Normalize Valuations | // Scale valuations so that $\mu_i^n(M) = 1, \forall i$ |
| 2 $(N, M, V) \leftarrow \text{Initial-Assignment}(N, M, V, \alpha)$ | // Algorithm 5 |
| 3 Bag-Filling(N, M, V, α) | // Algorithm 6 |
-

be checked by using the total valuation of low-value items. If the upper bounds are not tight enough for some agents, then we update the MMS value of such an agent and repeat. We show that we do not need to update the MMS upper bounds more than $O(n^3)$ times before we have a good upper bound on all MMS values. Then, we show that the same bag-filling procedure, as in the basic algorithm, satisfy every remaining agent. The running time of the entire algorithm is $O(nm(n^4 + \log m))$.

In Section 3.4, we show that our basic algorithm also yields a better bound. We show the existence of a $(\frac{3}{4} + \frac{1}{12n})$ -MMS allocation. The entire algorithm remains exactly the same but with an involved analysis. The analysis is tricky in this case, so we add a set of *dummy items* to make proofs easier. We use these items to make up for the extra loss for the remaining agents due to the additional factor, and, of course, these items are not assigned to any agent in the algorithm.

3.2 Existence of 3/4-MMS Allocation

In this section, we present a simple proof for the existence of a $\frac{3}{4}$ -MMS allocation for a given instance $\mathcal{I} = \langle N, M, V \rangle$. We assume that the MMS value μ_i of each agent i is given. Finding the exact μ_i is an NP-Hard problem, however a PTAS exists [48]. This implies a PTAS to compute a $(\frac{3}{4} - \epsilon)$ -MMS allocation for any $\epsilon > 0$. Using the properties stated in Section 2.3, we normalize valuations so that $\mu_i^n(M) = 1, \forall i$ (Lemma 2.5) and assume that $\mathcal{I}$ is an ordered instance, i.e., $v_{i1} \geq \dots \geq v_{im}, \forall i$ (Lemma 2.10). Our proof is algorithmic. Whenever we apply a valid reduction (Definition 2.7) and more than one agent satisfies the conditions (2.1), we choose one arbitrarily.

For the ease of exposition, we abuse notation and use M and N to respectively denote the set of unallocated items and the set of agents who have not received any bundle yet. We also use $n := |N|$ and $m := |M|$. Further, we use j to denote the j^{th} highest value item in M .

The algorithm is given in Algorithm 4. We use $\alpha = \frac{3}{4}$ in this section. Later, in Section 3.4, we use the same algorithm for $\alpha = \frac{3}{4} + \frac{1}{12n}$. Algorithm 4 has two main parts: Initial Assignment and Bag-Filling. We now describe each part separately in detail.

3.2.1 Initial Assignment

We first assign *high-value items* using Algorithm 5. We note that handling high-value items is the biggest challenge in the MMS problem, e.g., a major part of Ghodsi et al. algorithm [31] is devoted to allocating high-value items. In our algorithms, we make it a simple process of greedy assignment by leveraging the pigeonhole principle to make valid reductions. We define bundles $S_1 := \{1\}$, $S_2 := \{n, n+1\}$ ($\emptyset$, if $m \leq n$), $S_3 := \{2n-1, 2n, 2n+1\}$ ($\emptyset$, if $m \leq 2n$), and $S_4 := \{1, 2n+1\}$ ($\emptyset$, if $m \leq 2n$), where S_1 has the highest value item in M , S_2 has the n^{th} and $(n+1)^{\text{th}}$ highest valued items in M , and so on. We will show that allocating any of these bundles to an agent who values it at least $\frac{3}{4}$ is a valid reduction. Note that these

Algorithm 5: Initial-Assignment

Input : Ordered Instance $\mathcal{I} = \langle N, M, V \rangle$, where $\mu_i^n(M) = 1, \forall i \in N$, and an approximation factor α
Output : Reduced Instance

- 1 For any $S \subseteq M$, define $\Gamma(S) := \{i \in N : v_i(S) \geq \alpha\}$
- 2 $n \leftarrow |N|$ // n changes with N
- 3 $S_1 \leftarrow \{1\}; S_2 \leftarrow \{n, n+1\}; S_3 \leftarrow \{2n-1, 2n, 2n+1\}; S_4 \leftarrow \{1, 2n+1\}$ // bundles that can be assigned
- 4 **while** $(\Gamma(S_1) \cup \Gamma(S_2) \cup \Gamma(S_3) \cup \Gamma(S_4)) \neq \emptyset$ **do**
- 5 $S \leftarrow$ the lowest index bundle in $\{S_1, S_2, S_3, S_4\}$ for which $\Gamma(S) \neq \emptyset$
- 6 $i \leftarrow$ an agent in $\Gamma(S)$
- 7 Assign S to agent i // initial assignment
- 8 $M \leftarrow M \setminus S; N \leftarrow N \setminus \{i\}$
- 9 **return** $\langle N, M, V \rangle$

bundles change after every valid reduction.

In Algorithm 5, we keep assigning the lowest index $S \in \{S_1, S_2, S_3, S_4\}$ to agent i , if any, for which $v_i(S) \geq \frac{3}{4}$. Then, we update M and N to reflect the current unallocated items and agents who have not been assigned with any bundle yet. The following lemma extends the ideas that appeared in [31, 28].

Lemma 3.1. *Let S be the lowest index bundle in $S \in \{S_1, S_2, S_3, S_4\}$ for which $\Gamma(S) := \{i \in N : v_i(S) \geq \frac{3}{4}\}$ is non-empty. Then, removing S and agent i with $v_i(S) \geq \frac{3}{4}$ is a valid reduction.*

Proof. Clearly, $v_i(S) \geq \frac{3}{4}$. Therefore, we only need to show the second condition in Definition 2.7. We show this separately for each case of $S \in \{S_1, S_2, S_3, S_4\}$. Fix agent $i' \in N \setminus \{i\}$ and $P \in \mathcal{P}_{i'}^n(M)$ (Recall from Definition 2.1 that $\mathcal{P}_{i'}^n(M)$ denote the set of partitions achieving $\mu_{i'}^n(M)$). We show that after removing S , there exists a partition of $M \setminus S$ into $(n-1)$ bundles such that the value of each bundle is at least $\mu_{i'}^n(M)$, i.e., $\mu_{i'}^{n-1}(M \setminus S) \geq \mu_{i'}^n(M)$.

- $S = S_1$. Removal of one item from P affects exactly one bundle and each of the remaining $(n-1)$ bundles has value at least $\mu_{i'}^n(M)$. Therefore, $\mu_{i'}^{n-1}(M \setminus \{1\}) \geq \mu_{i'}^n(M), \forall i' \in N \setminus \{i\}$.
- $S = S_2$. In P , there exists a bundle with two items from $\{1, \dots, n+1\}$ (pigeonhole principle). Let T be a bundle in P that has two items from $\{1, \dots, n+1\}$. Let us exchange these items with items n and $n+1$ in other bundles and distribute any remaining items in T among other bundles arbitrarily. Clearly, the value of other bundles except T does not decrease, and hence $\mu_{i'}^{n-1}(M \setminus \{n, n+1\}) \geq \mu_{i'}^n(M), \forall i' \in N \setminus \{i\}$.
- $S = S_3$. Similar to the proof of Case ' $S = S_2$ '.
- $S = S_4$. In each iteration, the lowest index bundle from $\{S_1, S_2, S_3, S_4\}$ is picked. Therefore, S_4 is only picked when $v_i(S_1), v_i(S_3) < \frac{3}{4}$ for all $i \in N$ which implies that $v_{i'1} < \frac{3}{4}$ and $v_{i'(2n+1)} < \frac{1}{4}$ and hence $v_i(S_4) < 1$ for all $i \in N$.

In P , if items 1 and $2n+1$ are in the same bundle, then clearly, removing S_4 and agent i is a valid reduction. For the other case, if 1 and $2n+1$ are in two different bundles, then we can make two new bundles, one with $\{1, 2n+1\}$ and another with all the remaining items of the two bundles. The value of the bundle without $\{1, 2n+1\}$ is at least $2\mu_i - 1 \geq \mu_i$ because $v_i(S_4) < 1$ for all $i \in N$ and $\mu_i \geq 1$. Hence, this is a valid reduction. $\square$

Lemma 5.9 implies the following corollary.

Algorithm 6: Bag-Filling Algorithm for α -MMS Allocation

Input : Reduced Instance $\mathcal{I} = \langle N, M, V \rangle$ with $\mu_i^n = 1 \ \forall i$ and an approximation factor α

Output : Allocation $A = (A_1, \dots, A_n)$ where $v_i(A_i) \geq \alpha$

```
1 Initialize Bags  $B = \{B_k\}_{k \in [n]}$  as in (3.2) // see Figure 3.1
2  $R \leftarrow M \setminus J$ 
3 for  $k = 1$  to  $n$  do
4    $T \leftarrow B_k$ 
5   Define  $\Gamma(T) := \{i \in N : v_i(T) \geq \alpha\}$ 
6   while  $\Gamma(T) = \emptyset$  do
7      $j \in R$  // pick one low-value item arbitrarily
8      $T \leftarrow T \cup \{j\}; R \leftarrow R \setminus \{j\}$  // add the new item to bag
9    $i \in \Gamma(T); A_i \leftarrow T; N \leftarrow N \setminus \{i\}$  // assign  $T$  to  $i \in \Gamma(T)$ 
```

Corollary 3.2. *After the execution of Algorithm 5 is completed, $\mu_i \geq 1$ for all $i \in N$.*

Note that the original MMS value is 1 for each agent. Although it may increase after a valid reduction, we only need to give each agent a bundle of value at least $\frac{3}{4}$ to achieve a $\frac{3}{4}$ -MMS allocation for the original instance. For this reason, we are interested in partitions where the value of each bundle is at least 1 for an agent. Therefore, we abuse notation to denote by $\mathcal{P}_i^n(M)$ the set of partitions achieving the original MMS value, i.e.,

$$\mathcal{P}_i^n(M) = \{P \in \Pi_n(M) : \min_{P_k \in P} v_i(P_k) \geq 1\} . \quad (3.1)$$

Observe that $\mathcal{P}_i^n(M)$ contains MMS partitions of agent i for the current M and n as well as the partitions that do not achieve the current μ_i , but the value of each bundle is at least 1.

3.2.2 Bag-Filling

We use the bag-filling procedure given in Algorithm 6 to satisfy the remaining agents. Let $J_1 := \{1, \dots, n\}$ denote the set of first n items. Similarly, let us define $J_2 := \{n+1, \dots, 2n\}$ and $J := J_1 \cup J_2$. We call J to be the set of high-value items. The following corollary is straightforward.

Corollary 3.3. *If $v_i(S) < \frac{3}{4}$, for all i and for all $S \in \{S_1, S_2, S_3\}$, then (i) $v_{ij} < \frac{3}{4}$, $\forall j \in J_1$, (ii) $v_{ij} < \frac{3}{8}$, $\forall j \in J_2$, and $v_{in} < \frac{3}{4} - v_{i(n+1)}$, and (iii) $v_{ij} < \frac{1}{4}$, $\forall j \in M \setminus J$, for all i .*

Next, we initialize n bags as follows:

$$B = \{B_1, B_2, \dots, B_n\}, \text{ where } B_k = \{k, 2n - k + 1\}, \forall k . \quad (3.2)$$

Each bag contains one item from J_1 and one item from J_2 such that from B_1 to B_n value of items from J_1 decreases and value of items from J_2 increases (see Figure 3.1 for an illustration).

Algorithm 6 has n rounds. In each round k , it starts a new bundle T with $T \leftarrow B_k$. If there is an agent who values T to be at least $\frac{3}{4}$, then assign T to such an agent. Otherwise, keep adding items from $M \setminus J$ to T one by one until an agent with no bundle assigned to her values T at least $\frac{3}{4}$. The algorithm allocates T to that agent, and if there are multiple such agents, it chooses one arbitrarily.

For correctness, we need to show that there are enough items in $M \setminus J$ to add on top of each bag in (3.2)

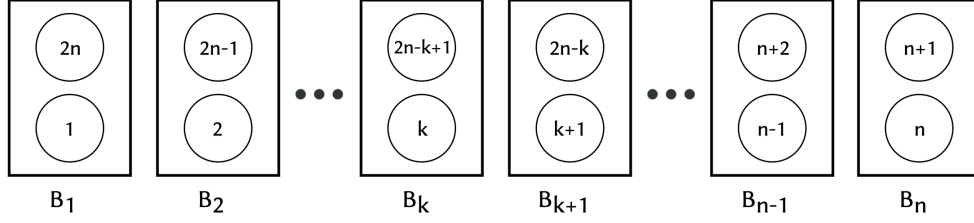


Figure 3.1: Setting of the items in the bags

so that each agent gets a bundle that they value at least $\frac{3}{4}$. For this, we first divide agents into two types:

$$N^1 := \{i \in N \mid v_i(B_k) \leq 1, \forall k\} \text{ and } N^2 := N \setminus N^1.$$

If N^2 is empty, then it is easy to check that using Corollary 3.3(iii) and the ideas in Section 2.3.1 that Algorithm 6 gives each agent at least $\frac{3}{4}$. We need some more notation to show correctness when N^2 is not empty.

For an agent $i \in N^2$, define

$$\begin{aligned} L_i &:= \{B_k : v_i(B_k) < \frac{3}{4}\}; & l_i &:= |L_i| \\ H_i &:= \{B_k : v_i(B_k) > 1\}; & h_i &:= |H_i| \\ x_i &:= (\frac{3}{4})l_i - \sum_{k: B_k \in L_i} v_i(B_k). \end{aligned} \tag{3.3}$$

In words, L_i is the set of bags that i values strictly less than $\frac{3}{4}$, H_i is the set of bags that i values strictly more than 1, and l_i and h_i are the number of bags in L_i and H_i respectively. Further, x_i is the least total value needed to make all bundles in L_i at least $\frac{3}{4}$. Agents in N^2 have nice properties that we show in the following lemma.

Lemma 3.4. *For an agent $i \in N^2$, (i) $l_i > 0$ and $h_i > 0$, (ii) $v_{i1} > \frac{5}{8}$, (iii) $v_i(B_k) < \frac{9}{8}, \forall k$, and (iv) $v_{ij} < \frac{1}{8}, \forall j \in M \setminus J$.*

Proof. For the first part, if $h_i = 0$ then $i \in N^1$. Further, $l_i = 0$ implies that $v_i(B_n) \geq \frac{3}{4}$, which cannot be true after the execution of Algorithm 5 due to Line 4-8 of the algorithm.

For the second part, $v_{ij} < \frac{3}{8}, \forall j \in J_2$ due to Corollary 3.3. Since $h_i > 0$, there exists a $j \in J_1$ such that $v_{ij} > \frac{5}{8}$. Further, since item 1 is the highest value item for every agent, $v_{i1} > \frac{5}{8}$.

For the third part, each item of J_1 has value less than $\frac{3}{4}$ and each item of J_2 has value less than $\frac{3}{8}$ (Corollary 3.3) for any agent. Therefore, $v_i(B_k) < \frac{9}{8}$ for any bundle B_k .

For the fourth part, the value of item 1 for an agent $i \in N^2$ is more than $\frac{5}{8}$ and $v_i(S_4) < \frac{3}{4}$, hence $v_{ij} \leq v_{i(2n+1)} < \frac{1}{8}, \forall j \in M \setminus J$. $\square$

In the following lemma, we show that if the total value of the items in $M \setminus J$ for each agent i in N^2 is at least $x_i + \frac{l_i}{8} - \frac{1}{8}$, i.e.,

$$v_i(M \setminus J) \geq x_i + \frac{l_i}{8} - \frac{1}{8},$$

then the bag-filling algorithm will assign every agent (in N^1 and N^2) a bundle with value at least $\frac{3}{4}$. In the rest of this section, we show that the bound on the value of $M \setminus J$ actually holds by using the fact that $\mu_i \geq 1$

for all $i \in N$.

Lemma 3.5. *If $v_i(M \setminus J) \geq x_i + \frac{l_i}{8} - \frac{1}{8}, \forall i \in N^2$, then Algorithm 6 gives every agent a bundle that they value at least $\frac{3}{4}$.*

Proof. This is proof by contradiction. Note that, in Algorithm 6, R is the set of unallocated items from $M \setminus J$ and T is the bag that is being filled at a time. Let $R^{(k)}$ and $T^{(k)}$ be respectively R in the beginning and T at the end of round k of the algorithm, i.e., $R^{(1)} = M \setminus J$ and $T^{(1)} \supseteq B_1$. For contradiction, suppose the algorithm stops at round t because there are not enough unallocated items in $R^{(t)}$ to satisfy any remaining agent i , i.e., $v_i(B_t \cup R^{(t)}) < \frac{3}{4}$.

If $i \in N^1$, each removed bundle in rounds $k \in [t-1]$, has value of at most 1 for agent i . Because, if $v_i(B_k) \geq \frac{3}{4}$ for $k \in [t-1]$, agent i is already interested in B_k ($i \in \Gamma(B_k)$) and the algorithm does not enter the while loop in Line 6. Therefore, if $v_i(B_k) \geq \frac{3}{4}$ no more item has been added to $T^{(k)} = B_k$. Also, if $v_i(B_k) < \frac{3}{4}$ for $k \in [t-1]$, before adding the last item (if any) to $T^{(k)}$, the value of $T^{(k)}$ is less than $\frac{3}{4}$ (otherwise, it would be out of loop and allocated to someone). Moreover, from Corollary 3.3, $v_{ij} < \frac{1}{4}$ for $j \in R^{(k)}$ (since $R^{(k)} \subseteq M \setminus J$). Therefore, at the end of the round k , $v_i(T^{(k)}) < 1$. Further, since $v_i(M) \geq n$ and $v_i(B_k) \leq 1, \forall k$, we have $v_i(B_t \cup R^{(t)}) \geq 1$, which is a contradiction.

If $i \in N^2$, then since at round t , $v_i(T^{(t)}) < \frac{3}{4}$, we have $B_t \in L_i$ (See (3.3) for the definition of L_i). Consider a round $k \in [t-1]$. If $B_k \notin L_i$, then $T^{(k)} = B_k$ has been assigned to someone with no additional items added to $T^{(k)}$ from $R^{(k)}$ because $i \in \Gamma(T)$ and the algorithm does not enter the while loop in Line 6. If $B_k \in L_i$, then in round k , before adding the last item (if any) to $T^{(k)}$, the value of i for $T^{(k)}$ is less than $\frac{3}{4}$. Moreover, from Lemma 3.4, each item in $R^{(k)}$ has value of at most $\frac{1}{8}$. Therefore, if $B_k \in L_i$, the value of the assigned bag for i in round k is less than $\frac{7}{8}$. Since $B_t \in L_i$, at most $l_i - 1$ bags from L_i have been assigned up to $t-1$ iterations. Further, since items from $M \setminus J$ are added to bags in L_i only, the total value taken from $M \setminus J$ up to $t-1$ iterations, according to agent i , is at most $x_i - (\frac{3}{4} - v_i(B_t)) + (l_i - 1)/8$ where $x_i - (3/4 - v_i(B_t))$ to make each of $L_i \setminus B_t$ exactly $\frac{3}{4}$ (See (3.3) for the definition of x_i) and $(l_i - 1)/8$ to add an extra $\frac{1}{8}$ to each. Hence, in the beginning of round t ,

$$v_i(R^{(t)}) \geq (x_i + \frac{l_i}{8} - \frac{1}{8}) - (x_i - (\frac{3}{4} - v_i(B_t)) + (l_i - 1)/8) = \frac{3}{4} - v_i(B_t), \quad (3.4)$$

which is a contradiction. $\square$

Now, we only need to show that for each $i \in N^2$, we have

$$v_i(M \setminus J) \geq x_i + \frac{l_i}{8} - \frac{1}{8}. \quad (3.5)$$

Showing (3.5)

In fact, we will show a stronger bound without $-\frac{1}{8}$ in Theorem 3.15. We will use the extra $-\frac{1}{8}$ to improve the bound in Section 3.4. We start with a few lemmas to show more properties of agents in N^2 , in addition to previous properties shown in Lemma 3.4. Recall from (3.1) that $\mathcal{P}_i^n(M)$ denote the set of partitions where the value of each bundle is at least 1.

Lemma 3.6. *For an agent $i \in N^2$, there exists a bundle P_k in every partition $P = \{P_1, \dots, P_n\} \in \mathcal{P}_i^n(M)$ such that $v_i(P_k \setminus J) > \frac{1}{4}$.*

Proof. If there exists a bundle P_k with exactly one item j from J , then $v_i(P_k \setminus J) \geq 1 - v_{ij} > \frac{1}{4}$ because the value of every item is less than $\frac{3}{4}$. Otherwise, each bundle has exactly two items from J , which implies that

one of the bundles, say P_k , has two items j_1, j_2 from the set $\{n\} \cup J_2$. Since $v_{ij_1} + v_{ij_2} \leq v_{in} + v_{i(n+1)} < \frac{3}{4}$, $v_i(P_k \setminus J) > \frac{1}{4}$. $\square$

Next, we show that there exists a partition in $\mathcal{P}_i^n(M)$ for $i \in N^2$ where all items with value more than $\frac{5}{8}$ are in separate bundles. The intuition of this proof is that the bundle which has two items of value greater than $\frac{5}{8}$ can be merged with another bundle (or possibly two other bundles) and make two (or three) bundles each value at least 1. This basically utilizes the extra $\frac{2}{8}$ value in the first bundle to reshuffle items to obtain the desired partition. We begin with the following claim.

Claim 3.7. *If there exists a partition $P = \{P_1, \dots, P_n\} \in \mathcal{P}_i^n(M)$ for an agent $i \in N^2$ where a bundle $P_k \in P$ contains two items with value more than $\frac{5}{8}$ for agent i . Then,*

1. *there exists another bundle $P_{k'} \in P$ for which $\max_{j \in P_{k'}} v_{ij} < \frac{3}{8}$.*
2. *if $v_i(P_k \setminus J) > \frac{1}{4}$, then we can make two new bundles from the items in $P_k \cup P_{k'}$ where each bundle has one item with value more than $\frac{5}{8}$ and each bundle values at least 1.*
3. *if $P_{k'}$ cannot be divided into two parts with value at least $\frac{3}{8}$ each. Then, $v_i(P_{k'} \setminus J) < \frac{1}{4}$.*

Proof. Corollary 3.3 implies that the only items that can have value more than $\frac{5}{8}$ are items in $J_1 = \{1, \dots, n\}$. Therefore, P_k has two items from J_1 . By the pigeonhole principle, there must exist another bundle $P_{k'}$ where all items are from $\{n+1, \dots, m\}$. Corollary 3.3 implies that each item in $P_{k'}$ has value less than $\frac{3}{8}$. This proves the first part.

For the second part, if $v_i(P_k \setminus J) > \frac{1}{4}$ we have $v_i(P_k) > \frac{5}{8} + \frac{5}{8} + \frac{1}{4} = \frac{3}{2}$. We make two bundles by initializing two empty bags and adding items from $P_k \cup P_{k'}$ one by one in decreasing order of value to the bundle which has the lower value. We will get new bundles with values v_1 and v_2 where $v_1 + v_2 > \frac{5}{2}$ and $|v_1 - v_2| < \frac{3}{8}$. The second inequality holds because each item in $P_{k'}$ and $P_k \setminus J$ has value less than $\frac{3}{8}$. This proves the second part.

For the third part, assume $v_i(P_{k'} \setminus J) \geq \frac{1}{4}$ for a contradiction. We initialize two empty bundles and add items from $P_{k'}$ one by one in decreasing order of value to the bundle with lower value. We get two new bundles with value v_1 and v_2 where $v_1 + v_2 = v_i(P_{k'}) \geq 1$ and $|v_1 - v_2| < \frac{1}{8}$, which implies $v_1, v_2 \geq \frac{3}{8}$. The second inequality holds because each item in $P_{k'} \setminus J$ has value less than $\frac{1}{8}$. $\square$

Lemma 3.8. *For every $i \in N^2$, there exists a partition $P = \{P_1, \dots, P_n\} \in \mathcal{P}_i^n(M)$ such that each $P_k \in P$ has at most one item j with $v_{ij} > \frac{5}{8}$.*

Proof. If there are more than two items with value more than $\frac{5}{8}$ in a bundle P_k of $P \in \mathcal{P}_i^n(M)$, then we add one of these items to $P_{k'}$, defined in Claim 3.7. This will ensure that the value of both P_k and $P_{k'}$ is at least 1. By repeating this, we can obtain a $P \in \mathcal{P}_i^n(M)$ that has at most two items with value more than $\frac{5}{8}$.

Next, we show that if there are two items j_1, j_2 each with value more than $\frac{5}{8}$ for an agent i in a bundle P_k of $P \in \mathcal{P}_i^n(M)$, then we can construct another $P' \in \mathcal{P}_i^n(M)$ where this is not true. Let $P_{k'} \in P$ be a bundle for which $\max_{j \in P_{k'}} v_{ij} < \frac{3}{8}$ (see Claim 3.7(1) for the proof of its existence).

Case 1: If $v_i(P_k \setminus J) > \frac{1}{4}$, using Claim 3.7(2), we make two bundles with value at least 1 and exactly one item with value more than $\frac{5}{8}$ in each.

Case 2: If $v_i(P_k \setminus J) \leq \frac{1}{4}$ and there exists a partition $Q_{k'}^1$ and $Q_{k'}^2$ of items in $P_{k'}$ such that value of each $Q_{k'}^1$ and $Q_{k'}^2$ is at least $\frac{3}{8}$, then we can rearrange items in $P_k \cup P_{k'}$ and make two new bundles $(\{j_1\} \cup Q_{k'}^1)$ and $(\{j_2\} \cup Q_{k'}^2)$. Clearly, the value of each bundle is at least 1 and each has exactly one item with value more than $\frac{5}{8}$.

Case 3: Finally, if $v_i(P_k \setminus J) \leq \frac{1}{4}$ and no such $Q_{k'}^1$ and $Q_{k'}^2$ exists (as in Case 2), then we claim that there exists a partition of $P_{k'}$ into three bundles, each with value less than $\frac{3}{8}$. We can find this partition as follows: Initialize three empty bundles, and repeatedly add the highest value item of $P_{k'}$ to the bundle with the lowest value. For a contradiction, suppose one of the three bundles has value more than $\frac{3}{8}$, then the sum of the values of the other two sets must be less than $\frac{3}{8}$ because otherwise, they make a partition of two where each has value more than $\frac{3}{8}$. This means that at least one of the bundles must have a value less than $\frac{3}{16}$. This implies that the value of the largest bag before adding the last item must be less than $\frac{3}{16}$ and the last item also should value less than $\frac{3}{16}$, which is a contradiction. Therefore, there exists a partition $Q_{k'}^1, Q_{k'}^2$, and $Q_{k'}^3$ of $P_{k'}$ such that each has value less than $\frac{3}{8}$.

According to Lemma 3.6, there exists a bundle $P_{\hat{k}} \in \mathcal{P}_i^n(M)$ such that $v_i(P_{\hat{k}} \setminus J) > \frac{1}{4}$. Let $v_i(P_{\hat{k}}) = 1 + \delta$ for some $\delta \geq 0$. $P_{\hat{k}}$ cannot be same as P_k (Case 1 of this proof) and $P_{k'}$ (Claim 3.7(3)). We initialize three bags:

$$\text{bag 1 : } \{j_1\} \cup Q_{k'}^1 \quad \text{bag 2 : } \{j_2\} \cup Q_{k'}^2 \quad \text{bag 3 : } (P_k \setminus \{j_1, j_2\}) \cup (P_{\hat{k}} \cap J) \cup Q_{k'}^3$$

Observe that the value of each of bag 1 and bag 2 is at most $\frac{9}{8}$, and the total value of all items in $P_k, P_{k'}$ and $P_{\hat{k}}$ is at least $3 + \frac{2}{8} + \delta$. We sort the remaining items in decreasing order and add them one by one to a bag with the lowest value. Since the value of the last item added is at most $\frac{1}{8}$ (Lemma 3.4), each bag has a value of at least 1 and it has at most one item with value more than $\frac{5}{8}$. We repeat this process for each bundle with two items of value more than $\frac{5}{8}$ to find a desired partition. $\square$

Let a be an agent in N^2 . For simplicity, until the end of this section, when we use *value* of an item or a bundle, we mean the value for agent a (unless mentioned otherwise). Recall that we need to show (3.5). Let $P = (P_1, \dots, P_n) \in \mathcal{P}_a^n(M)$ be a partition satisfying the condition of Lemma 3.8, i.e., there is at most one item with value greater than $\frac{5}{8}$ in P_k for all $k \in [n]$.

Using (3.2), we manipulate P as follows: First, for each $B_i = \{j, j'\}$ with $\frac{3}{4} \leq v_a(B_i) \leq 1$, if $j \in P_k$ and $j' \in P_{k'}$, then we turn P_k and $P_{k'}$ into two new bundles $\{j, j'\}$ and $((P_k \cup P_{k'}) \setminus \{j, j'\})$. Observe that $v_a((P_k \cup P_{k'}) \setminus \{j, j'\}) \geq 1$. Hence, we can assume that for each $B_i \notin L_a \cup H_a$ (as defined in (3.3)), there exists a $P_k = B_i$ and all other bundles value at least 1. Second, we re-enumerate the bundles in P such that

- $P_1, \dots, P_{t_1}$: each has an item j of value more than $\frac{5}{8}$ and $v_a(B_j) < \frac{3}{4}$.
- $P_{t_1+1}, \dots, P_{t_2}$: each has an item j of value more than $\frac{5}{8}$ and $v_a(B_j) > 1$ (Observe that there are h_a such bundles, so $t_2 - t_1 = h_a$).
- $P_{t_3+1}, \dots, P_n$: each such $P_k = B_{k'}$ for some k' and $\frac{3}{4} \leq v_a(B_{k'}) \leq 1$.
- $P_{t_2+1}, \dots, P_{t_3}$ be the remaining bundles (Observe that $t_3 = l_a + h_a$).

Let $L_a^1 \subseteq L_a$ be the set of bags in L_a , which have one item with value more than $\frac{5}{8}$, and $L_a^2 = L_a \setminus L_a^1$. Clearly, $|L_a^1| = t_1$ and $|L_a^2| = l_a - t_1$.

Observation 3.9. 1. If $v_a(B_j) > 1 \geq v_{aj} > \frac{5}{8}$, then by Corollary 3.3, $v_{a(2n-j+1)} > \frac{1}{4}$.

2. If $v_a(B_j) < \frac{3}{4}$ and $v_{aj} > \frac{5}{8}$, i.e., $B_j \in L_a^1$ then $v_{a(2n-j+1)} < \frac{1}{8}$.

3. Using the two previous observations, if $B_{j_1} \in L_a^1$ and $B_{j_2} \in H_a$, then $j_1 < j_2$ because $v_{a(2n-j_2+1)} > \frac{1}{4} > \frac{1}{8} > v_{a(2n-j_1+1)}$.

4. if $B_{j_1} \in H_a$ and $B_{j_2} \in L_a^2$, then $j_1 < j_2$ because $v_{aj_1} > \frac{5}{8} \geq v_{aj_2}$.

5. Bags are ordered by value of most-valued item in each bag. First, L_a^1 then H_a then L_a^2 (excluding the bags for which $\frac{3}{4} \leq v_a(B_k) \leq 1$).
6. The above observations imply that items with value in $[\frac{1}{8}, \frac{1}{4}]$ (if any) automatically belong to bags B_j 's, which are neither in H_a nor L_a . Hence, there is no such item in $\{P_1, \dots, P_{t_3}\}$.
7. After the termination of Algorithm 5 we have $v_i(S_2) = v_i(B_n) < \frac{3}{4}$. Therefore, $B_n \in L_a$. Further, since $h_a > 0$ (Lemma 3.4), $B_n \in L_a^2$. This implies that $|L_a^2| \geq 1$.

Further, observe that there is no item from $M \setminus J$ in $P_{t_3+1}, \dots, P_n$. We leverage this observation to prove (3.5) by only considering partitions $\{P_1, \dots, P_{t_3}\}$ that contain all items of $M \setminus J$. Here is a simple example to help understand the construction.

Example 3.10. Consider an example where, after the execution of Algorithm 5, $n = 6$ (hence $|J| = 12$). The valuation of agent a for items in J is shown in Table 3.1. The construction of bags is shown in Figure 3.2 and Table 3.2.

Table 3.1: Valuation of agent a for items in J in Example 3.10

Item	1	2	3	4	5	6	7	8	9	10	11	12
v_{aj}	$\frac{28}{40}$	$\frac{28}{40}$	$\frac{28}{40}$	$\frac{28}{40}$	$\frac{17}{40}$	$\frac{15}{40}$	$\frac{14}{40}$	$\frac{14}{40}$	$\frac{14}{40}$	$\frac{13}{40}$	$\frac{5}{40}$	$\frac{1}{40}$

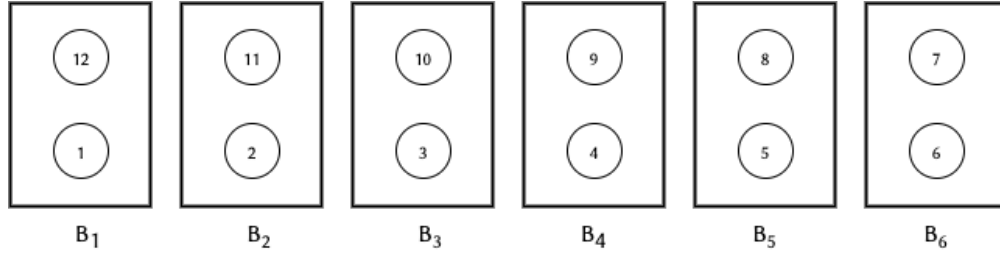


Figure 3.2: Setting of the items in the bags in Example 3.10

Table 3.2: Bags and their valuations in Example 3.10 where $(H_a \cup L_a)^c$ refers to the bags not in $(H_a \cup L_a)$.

Bag	B_1	B_2	B_3	B_4	B_5	B_6
Setting	$\{1, 12\}$	$\{2, 11\}$	$\{3, 10\}$	$\{4, 9\}$	$\{5, 8\}$	$\{6, 7\}$
$v_a(B_k)$	$\frac{29}{40}$	$\frac{33}{40}$	$\frac{41}{40}$	$\frac{42}{40}$	$\frac{31}{40}$	$\frac{29}{40}$
Bag type	L_a^1	$(H_a \cup L_a)^c$	H_a	H_a	$(H_a \cup L_a)^c$	L_a^2

Now let P be an MMS partition of agent a satisfying the condition of Lemma 3.8 be as Figure 3.3. $M_k \subseteq (M \setminus J)$ is the set of low-value items in P_k so $\bigcup_{k \in [6]} M_k = M \setminus J$. M_k 's can be possibly empty. Then, the enumerated P is shown in Figure 3.4. Observe that:

- $v_a(B_2) \in [\frac{3}{4}, 1]$. Therefore, items 2 and 11 are put into one bundle (P_5) and all remaining items, which were previously in the same bundle with each of these items, are in another bundle (P_4). This is the case for B_5 and items 5 and 8 too. These bundles are put in the last (P_5 and P_6).
- $B_1 \in L_a^1$ and also $v_{a1} > \frac{5}{8}$. Therefore, we set the bundle containing item 1 to be P_1 . B_1 is the only bag with this property and hence $t_1 = 1$.

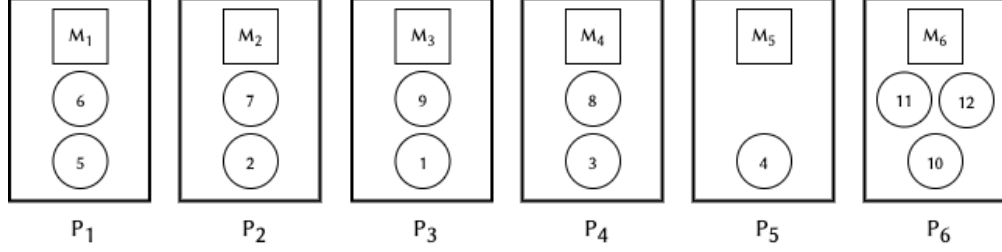


Figure 3.3: an MMS partition (P) of agent a in Example 3.10

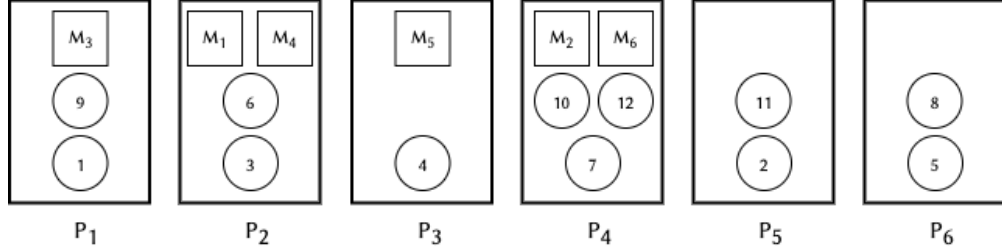


Figure 3.4: Re-enumerated P in Example 3.10

- $B_3, B_4 \in H_a$. Therefore, we put the bundles containing item 3 and item 4 next (P_2 and P_3). Therefore, $t_2 = 3$.
- The remaining bundle is set to be next (P_4). It means that $t_3 = 4$.

Definition 3.11. Define $y := v_{ak'}$ where k' is the highest value item in the set $\{j \in J_2 \cap B_k, B_k \in H_a\}$.

Observe that y is the value of the highest valued item from J_2 (items in top row of Figure 3.1), which is in a bag with value more than 1. Further, $\frac{2}{8} < y < \frac{3}{8}$ since by definition $y + v_{ij} > 1$ for some $j \in J_1$ and $y = v_{ij'}$ for some $j' \in J_2$. For instance, in Figure 3.1, if all bags in $\{B_{k+1}, \dots, B_n\}$ value at most 1 for agent a and $v_a(B_k) > 1$, i.e., B_k is in H_a , then $y = v_{a(2n-k+1)}$. This also implies that $v_{a1} \geq \dots \geq v_{ak} > \frac{5}{8}$. Moreover, $y = v_{a(2n-k+1)} \leq v_{a(2n-k)} \leq \dots \leq v_{a(k+1)}$. In Example 3.10, $y = \frac{14}{40}$.

The value y turns out to be crucial, which we use to relate the total value of $\cup_{k=1}^{t_3} P_k \cap (M \setminus J)$ to the value needed in bag-filling procedure. Intuitively, if y is high, then less value is needed from $M \setminus J$ to make all bundles in $\{P_1, \dots, P_{t_3}\}$ to be at least 1. This is also true for the bag-filling argument: If y is high then less value is needed to add to the bags in (3.2) to make each of them at least $\frac{3}{4}$. Otherwise, when y is low, the value of $M \setminus J$ can be shown to be relatively high. We begin with the following lemma.

Lemma 3.12. For any bundle $B_k = \{j, j'\} \in L_a$ such that $v_{aj}, v_{aj'} < \frac{5}{8}$, we have $v_{aj}, v_{aj'} \geq y$.

Proof. The claim follows from the construction of B_k 's, and the fact that each bundle of H_a has an item with value more than $\frac{5}{8}$. Let k be the index of the bundle in (3.2) for which the y in Definition 3.11 is achieved. Then

$$v_{a1} \geq \dots \geq v_{ak} > \frac{5}{8} \geq v_{a(k+1)} \geq \dots \geq v_{a(2n-k+1)} = y \geq v_{a(2n-k+2)} \geq \dots \geq v_{a(2n)}.$$

If $v_{aj} < \frac{5}{8}$ for $j \in J_1$, then $j > k$ and $j' < 2n - k + 1$, implying the result. $\square$

First, we only consider the case when bags in L_a satisfy Lemma 3.12. This is equivalent to the case when $t_1 = 0$. Then, we prove the general case in Theorem 3.15.

In the bags, which are in H_a , there are exactly $2h_a$ items from J . However, in $\{P_{t_1+1}, \dots, P_{t_2}\}$, we know that each bundle has one item with value $> \frac{5}{8}$ but may contain more items from J . We define z to reflect the possible difference, i.e., $z := \max\{2h_a - |\bigcup_{t=t_1+1}^{t_2} P_t \cap J|, 0\}$. In Lemma 3.13, we show two things. First we show a bound for the value that is needed with respect to y to make bags in L_a to become at least $\frac{3}{4}$ (see definition of x_i in (3.3)). Second, we show a bound for $M \setminus J$ with respect to z and y by using the fact that the value of each bundle in $\{P_1, \dots, P_{t_3}\}$ is at least 1. As we discussed before, we will relate these two observations to prove the bound for $v_a(M \setminus J)$ and we show (3.5) is always correct.

Lemma 3.13. *For agent $a \in N^2$, if $t_1 = 0$ then we have*

$$1. x_a \leq (\frac{3}{4} - 2y)l_a.$$

$$2. v_a(M \setminus J) \geq \max\{x_a + \frac{l_a}{4} - yz + \frac{z}{4}, \frac{z}{4}\}, \text{ where } z = \max\{2h_a - |\bigcup_{t=t_1+1}^{t_2} P_t \cap J|, 0\}.$$

Proof. For the first part, since $t_1 = 0$, we have $v_a(B_k) \geq 2y, \forall B_k \in L_a$ (Lemma 3.12). This implies that $x_a = (\frac{3}{4})l_a - \sum_{k: B_k \in L_a} B_k \leq (\frac{3}{4} - 2y)l_a$.

For the second part, in $\bigcup_{k: B_k \in H_a} B_k$, there are exactly $2h_a$ items from J . If there are at least $2h_a$ items in $\bigcup_{t=t_1+1}^{t_2} P_t \cap J$ then $z = 0$. On the other hand, if there are $2h_a - z$ items in $\bigcup_{t=t_1+1}^{t_2} P_t \cap J$, then it implies that there are at least z bundles in $\{P_{t_1+1}, \dots, P_{t_2}\}$ with exactly one item from J . Each of these bundles need more than $\frac{1}{4}$ value of items from $M \setminus J$ to be 1.

Next, if $z = 0$, it means that there are at least $2h_a$ items from J in $\{P_{t_1+1}, \dots, P_{t_2}\}$ and $2l_a$ items from J in $\{P_{t_2+1}, \dots, P_{t_3}\}$ since $t_1 = 0$. Then all bundles in $\{P_{t_2+1}, \dots, P_{t_3}\}$ need at least $x_a + \frac{l_a}{4}$ value of items from $M \setminus J$ to become 1 because the value of each item in the bundles of L_a is at least y (follows from Lemma 3.12 and Observation 3.9). If $z > 0$, then all bundles in $\{P_{t_2+1}, \dots, P_{t_3}\}$ need at least $x_a + \frac{l_a}{4} - yz$ value of items from $M \setminus J$ to become one because each of the z items has value at most y (Lemma 3.12). Therefore, in total, there should be at least $x_a + \frac{l_a}{4} - yz + \frac{z}{4}$ value of items from $M \setminus J$ in $\{P_{t_1+1}, \dots, P_{t_3}\}$. Letting $Q = P_{t_2+1} \cup \dots \cup P_{t_3}$, we have (using $v(P_t) \geq 1$ and $t_3 - t_2 = l_a$, and abusing notation):

$$\begin{aligned} l_a &\leq v_a(Q) \\ &= v_a(Q \cap J) + v_a(Q \cap (M \setminus J)) \\ &= v_a(Q \cap H_a) + v_a(Q \cap L_a) + v(Q \cap (M \setminus J)) \\ &\leq zy + v_a(L_a) + v_a(Q \cap (M \setminus J)) \\ &= zy + (\frac{3}{4})l_a - x_a + v_a(Q \cap (M \setminus J)), \end{aligned}$$

implying $v_a(Q \cap (M \setminus J)) \geq x_a + \frac{l_a}{4} - yz$, where the second-to-last line uses the fact that $|Q \cap J| = 2l_a + z$, so obviously $v_a(Q \cap J) \leq v_a(L_a) + zy$ (any items from H_a that end up in Q are worth at most y , which is at most the value of any L_a item), and the last line follows from substituting $x_a = (\frac{3}{4})l_a - v_a(L_a)$.

Further, in case z is large such that the value of $x_a + \frac{l_a}{4} - yz + \frac{z}{4}$ is negative, we still have z bundles in $\{P_{t_1+1}, \dots, P_{t_2}\}$ with exactly one item (with value of at most $\frac{3}{4}$) from J in each. Therefore, we need $\frac{z}{4}$ value of items from $M \setminus J$ to make all bundles in $\{P_{t_1+1}, \dots, P_{t_2}\}$ value 1. $\square$

In Lemma 3.13, we showed that $v_a(M \setminus J) \geq \max\{x_a + \frac{l_a}{4} - yz + \frac{z}{4}, \frac{z}{4}\}$. In Theorem 3.14, we prove (3.5) (for the special case when $t_1 = 0$) by relating the two observations from Lemma 3.13 and using two different cases for $\max\{x_a + \frac{l_a}{4} - yz + \frac{z}{4}, \frac{z}{4}\}$.

Theorem 3.14. For agent $a \in N^2$, if $t_1 = 0$ then

$$v_a(M \setminus J) \geq x_a + \frac{l_a}{8} . \quad (3.6)$$

Proof. Using Lemma 3.13(2) there are two cases for the maximum of $x_a + \frac{l_a}{4} - yz + \frac{z}{4}$ and $\frac{z}{4}$. We prove the claim for each case separately.

Case 1: Suppose $x_a + \frac{l_a}{4} - yz + z/4 \geq z/4$, then $z \leq \frac{x_a + \frac{l_a}{4}}{y}$. Further, Lemma 3.13(1) implies $y \leq \frac{3l_a - 4x_a}{8l_a}$. Using these, we get

$$v_a(M \setminus J) \geq x_a + \frac{l_a}{4} - z(y - \frac{1}{4}) \geq \frac{4x_a + l_a}{16y} \geq \frac{(4x_a + l_a)l_a}{2(3l_a - 4x_a)} .$$

Next, consider

$$\frac{(4x_a + l_a)l_a}{2(3l_a - 4x_a)} - \left(x_a + \frac{l_a}{8}\right) = \frac{28x_a^2 + (2x_a - l_a)^2}{8(3l_a - 4x_a)} \geq 0 .$$

The last inequality follows because $(3l_a - 4x_a) > 0$ (Lemma 3.4). This implies $v_a(M \setminus J) \geq x_a + \frac{l_a}{8}$.

Case 2: Suppose $x_a + \frac{l_a}{4} - yz + \frac{z}{4} \leq \frac{z}{4}$. Then, using Lemma 3.13(1), we have

$$v_a(M \setminus J) \geq \frac{z}{4} \geq \frac{4x_a + l_a}{16y} \geq \frac{(4x_a + l_a)l_a}{2(3l_a - 4x_a)} \geq x_a + \frac{l_a}{8} . \quad \square$$

Next, we handle the general case when $t_1 > 0$. Let $J_{\min} = \bigcup_{B_k \in L_a^1} B_k \cap J_2$ (recall that L_a^1 denote the set of bags in L_a that have one item with value more than $\frac{5}{8}$). Then, we have $v_{aj} < \frac{1}{8}, \forall j \in J_{\min}$ because these items are bagged with an item value more than $\frac{5}{8}$ and together they have value less than $\frac{3}{4}$. Therefore, $v_a(J_{\min}) < \frac{t_1}{8}$. Let us call items in $(M \setminus J) \cup J_{\min}$ as *filler* items (each filler item, by Lemma 3.4(iv), has value $< \frac{1}{8}$) and items in $J \setminus J_{\min}$ as *base* items.

Theorem 3.15. For agent $a \in N^2$, we have $v_a(M \setminus J) \geq x_a + \frac{l_a}{8}$.

Proof. In this proof we only consider the items in $\{P_1, \dots, P_{t_3}\}$ because, as discussed before, these bundles contain all items from $M \setminus J$. Let $x'_a := \frac{3}{4}t_1 - \sum_{B_k \in L_a^1} v_a(B_k)$, and $x''_a := x_a - x'_a$. For agent a , we can treat items of $J_{\min}$ as low-value items so we will prove:

$$v_a(M \setminus J) + v_a(J_{\min}) \geq x'_a + \frac{t_1}{4} + x''_a + (l_a - t_1)/8. \quad (3.7)$$

Since $v_a(J_{\min})$ is at most $\frac{t_1}{8}$, (3.7) directly implies the theorem.

In $\{P_1, \dots, P_{t_3}\}$, recall that each of $\{P_1, \dots, P_{t_1}\}$ contains a base item whose value is at least the maximum of all the items in the remaining bundles. Further, if there are two base items in any bundle P_k in $\{P_1, \dots, P_{t_1}\}$, then $v_a(P_k \cap J) > 1$ (using Observation 3.9). If each bundle in $\{P_1, \dots, P_{t_1}\}$ has exactly one base item (t_1 bundles in total) then items of value at least $x'_a + \frac{t_1}{4}$ are needed to make all these bundles at least 1. This, together with Theorem 3.14, proves the bound. On the other hand, if there are more than t_1 *base* items in $\{P_1, \dots, P_{t_3}\}$, by Observation 3.9(vi), each base item in $\{P_1, \dots, P_{t_3}\}$ has value $> \frac{1}{4}$. Therefore, for each extra base item j , there are two cases. If j comes from $B_k \in L_a^2$ then clearly, there should be enough items in $M \setminus J$ to compensate for v_{aj} in $\{P_{t_2+1}, \dots, P_{t_3}\}$ since the value item j is adding to a bag in L_a^1 is more than what it was considered in the right side of (3.7). For the other case, if j comes from $B_k \in H_a$ then it will

Algorithm 7: $\frac{3}{4}$ -MMS Allocation

Input : Ordered Instance $\langle N, M, V \rangle$, i.e., $v_{i1} \geq \dots \geq v_{im}, \forall i \in N$
Output : $\frac{3}{4}$ -MMS Allocation

```
1 Normalize Valuations (i.e., scale valuations so that  $v_i(M) = n, \forall i \in N$ )           //  $n = |N|$ 
2  $(N, M, V) \leftarrow \text{Fixed-Assignment}(N, M, V)$                                    // Algorithm 8
3  $(N^t, M^t, V^t) \leftarrow \text{Tentative-Assignment}(N, M, V)$                        // Algorithm 9
4 while  $N^{21} \neq \emptyset$  do
5    $i \leftarrow$  the lowest index agent in  $N^{21}$                                      //  $N^{21}$  is defined in (3.8)
6   Undo Tentative-Assignments                                                         // go back to  $(N, M, V)$ 
7   Update  $i$ 's MMS bound & normalize valuations                                     // Theorem 3.19
8    $(N, M, V) \leftarrow \text{Fixed-Assignment}(N, M, V)$                              // these assignments are final
9    $(N^t, M^t, V^t) \leftarrow \text{Tentative-Assignment}(N, M, V)$ 
10 Make all tentative assignments final:  $(N, M, V) \leftarrow (N^t, M^t, V^t)$ 
11 Bag-Filling( $N, M, V, \alpha$ ) for  $\alpha = \frac{3}{4}$                                      // Section 3.3.3 and Algorithm 6
```

make $B_k \setminus \{j\}$ to need more low-value items than the bundle in $\{P_1, \dots, P_{t_1}\}$ who gets j to become one since if $B_k \in H_a$ and $B_{k'} \in L_a^1$ then $v_{ak} < v_{ak'}$ (Observation 3.9(v)). This completes the proof. $\square$

3.3 Algorithm for $3/4$ -MMS Allocation

The existence proof of $\frac{3}{4}$ -MMS allocation in Section 3.2 requires the knowledge of the exact MMS value μ_i 's of all agents, e.g., the proof of valid reduction for bundle S_4 in Lemma 5.9 needs this assumption. Finding an exact μ_i of an agent i is an NP-Hard problem, however a PTAS exists [48]. This implies a PTAS to compute a $(\frac{3}{4} - \epsilon)$ -MMS allocation for an $\epsilon > 0$. However, for small ϵ , this PTAS is computationally very expensive and may not be practical. In this section, we show that our algorithmic technique in Section 3.2 is powerful enough to be modified into a strongly polynomial time algorithm to find an exact $\frac{3}{4}$ -MMS allocation for a given instance $\mathcal{I} = \langle N, M, V \rangle$. The key idea is to use the average as an upper-bound for the MMS value; see Lemma 2.4.

As it is shown in Section 2.3, we assume $\mathcal{I}$ to be an ordered instance, i.e., $v_{i1} \geq v_{i2} \geq \dots \geq v_{im}$, for all $i \in N$. We fix the order of agents in N such that at each step of the algorithm if more than one agent satisfies the conditions of valid reduction, defined in Section 2.3.2, we choose one with the lowest index. We maintain the same order among remaining agents after every valid reduction. For the ease of exposition, as in Section 3.2, we abuse notation and use M and N to denote the set of unallocated items and the set of agents who have not received any bundle yet, respectively. Moreover, we use $n := |N|$ and $m := |M|$. Further, we use j to denote the j^{th} highest value item in M , and i to denote the i^{th} agent in N as per the fixed order.

In this section, unlike what we did in Section 3.2 (normalizing the valuation of items based on actual $\mu_i^n(M)$), we normalize v_{ij} 's using the total value of all items.

Using the scale invariance property (Lemma 2.5), we can without loss of generality work with the normalized valuations so that $v_i(M) = n$ for all agents $i \in N$. The proof of following corollary is straightforward.

Corollary 3.16. *For a normalized instance $\mathcal{I} = \langle N, M, V \rangle$, $\mu_i^n(M) \leq 1 \ \forall i$.*

The algorithm to compute a $\frac{3}{4}$ -MMS allocation is given in Algorithm 7. It has three main parts: Initial Assignment, Update Upper Bound, and Bag-Filling. Initial Assignment is further divided into two parts: Fixed and Tentative.

Algorithm 8: Fixed-Assignment

Input : Ordered Instance $\langle N, M, V \rangle$ (i.e., $v_{i1} \geq \dots \geq v_{im}, \forall i \in N$), where $\mu_i^n(M) \leq 1, \forall i \in N$

Output : Fixed Assignments and Reduced Instance

```
1 For any  $S \subseteq M$ , define  $\Gamma(S) = \{i \in N : v_i(S) \geq \frac{3}{4}\}$ 
2  $S_1 := \{1\}; S_2 := \{n, n+1\}; S_3 := \{2n-1, 2n, 2n+1\}$  // bundles that can be assigned
3  $T = \Gamma(S_1) \cup \Gamma(S_2) \cup \Gamma(S_3)$ 
4 while  $T \neq \emptyset$  do
5    $S \leftarrow$  the lowest index bundle in  $\{S_1, S_2, S_3\}$  for which  $\Gamma(S) \neq \emptyset$ 
6    $i \leftarrow$  the lowest index agent in  $\Gamma(S)$ 
7   Assign  $S$  to agent  $i$  // final assignment
8   Update  $N, M, V, T$ 
9 return  $N, M, V$ 
```

3.3.1 Initial Assignment

This section is very similar to Section 3.2.1 where we assign S_1, S_2, S_3 , and S_4 to agents in order to reduce the number of high-value items to at most $2n$. The main difference for this section is assigning S_4 . Since assigning S_4 is a *valid reduction* only if $\mu_i^n(M) = 1$ (Lemma 5.9), therefore, when S_1, S_2 , and S_3 value less than $\frac{3}{4}$ for all agents we check for S_4 and if the first S_4 is allocated, then this allocation and all initial allocations after that are called *tentative assignments*. Tentative assignment are finalized only if allocation of any S_4 did not take away too much valuation for any agent in N .

It is crucial for this step to assign bundles to agents according to an order. Therefore, for S_1, S_2, S_3 , and S_4 the bundle with lower index has more priority and if there are more than one agent satisfied with $S \in \{S_1, S_2, S_3, S_4\}$ we choose the one with lower index based on the original order of N . This ordering will be useful when we have to undo tentative assignments.

Update N, M, V After every assignment, say bundle S to agent i , we update N, M, V as follows:

$$M \leftarrow M \setminus S; \quad N \leftarrow N \setminus \{i\}; \quad v_{i'j} \leftarrow v_{i'j} \cdot \frac{|N|}{v_{i'}(M)}, \forall i' \in N, j \in M.$$

Note that this maintains $\mu_i^n(M) \leq 1, \forall i \in N$ after normalization (Definition 2.6 and Lemma 2.4).

Fixed Assignment

In this part, we allocate high-value items to agents using Algorithm 8. We assign the bundle $S \in \{S_1, S_2, S_3\}$ to the lowest index agent $i \in N$ for which $v_i(S) \geq \frac{3}{4}$. Then, we update N, M, V and repeat this step until no such agent exists. Lemma 5.9 is applicable here to show that allocating $S \in \{S_1, S_2, S_3\}$ to agent i and removing them from M and N is a *valid reduction*.

As in Section 3.2, let $J_1 := \{1, \dots, n\}$ denote the set of first n items. Similarly, define $J_2 := \{n+1, \dots, 2n\}$ and $J := J_1 \cup J_2$. Note that Corollary 3.3 is also applicable here.

Tentative Assignment

This step is a continuation of Fixed Assignment in Section 3.3.1. It starts when Algorithm 8 terminates and the first S_4 bundle is being assigned. In this part, we assume that $\mu_i^n(M) = 1, \forall i \in N$, i.e., the actual MMS value for each agent is equal to her current MMS upper bound, and we allocate S_1, S_2, S_3, S_4 using

Algorithm 9: Tentative-Assignment

Input : Ordered Instance $\langle N, M, V \rangle$ that satisfies Conditions 1-3 in Corollary 3.3

Output : Tentative Assignments and Reduced Instance

```
1 For any  $S \subseteq M$ , define  $\Gamma(S) = \{i \in N : v_i(S) \geq \frac{3}{4}\}$ 
2  $S_1 := \{1\}$ ;  $S_2 := \{n, n+1\}$ ;  $S_3 := \{2n-1, 2n, 2n+1\}$ ;  $S_4 := \{1, 2n+1\}$ 
3  $T = \Gamma(S_1) \cup \Gamma(S_2) \cup \Gamma(S_3) \cup \Gamma(S_4)$ 
4 while  $T \neq \emptyset$  do
5    $S \leftarrow$  the lowest index bundle in  $\{S_1, S_2, S_3, S_4\}$  for which  $\Gamma(S) \neq \emptyset$ 
6    $i \leftarrow$  the lowest index agent in  $\Gamma(S)$ 
7   Assign  $S$  to agent  $i$  // tentative assignment
8   Update  $N, M, V, T$ 
9 return  $N, M, V$ 
```

this assumption. In the next step, either this assumption works fine and we make all tentative assignments final and move to the next stage of bag-filling or we detect an agent i for whom $\mu_i^n(M)$ is significantly lower than 1. In the latter case, we update i 's MMS upper bound. In particular, we update the MMS upper bound by updating the valuations so that the new MMS upper bound remains 1 for every agent.

In Algorithm 9, for $S \in \{S_1, S_2, S_3, S_4\}$ we check whether there exists an $i \in N$ with $v_i(S) \geq \frac{3}{4}$. If true, then we *tentatively* assign the lowest index such a bundle S to the lowest index such an agent i and we *tentatively* update M, N, V as we did in Section 3.3.1. Choosing the lowest index S_k makes sure that when S_4 is assigned, then none of S_1, S_2, S_3 satisfies the condition. This is essential in proving that assigning S_4 to an agent is a valid reduction in Lemma 5.9. Further, note that when S_4 is assigned, the value of each bundle in $\{S_1, S_2, S_3\}$ for every agent is strictly less than $\frac{3}{4}$. However, after removing an agent i with S_4 , it may later trigger valid reductions with $\{S_1, S_2, S_3\}$.

3.3.2 Updating MMS Upper Bound

The goal here is first to detect the agents for whom the MMS upper bound is overestimated and second to update this bound for them. To detect these agents we continue as in Section 3.2 by first initializing n bags as in (3.2) (see Figure 3.1), and divide agents into two types N^1 and N^2 according to their valuations for these bags. Recall that N^1 is the set of agents whose value for each bag is at most 1, and N^2 is the set of remaining agents.

In this section, we need to analyze agents in N^2 more thoroughly. We further partition agents in N^2 into two sub-types as follows:

$$N^{21} := \{i \in N^2 : h_i > l_i \text{ and } v_i(M \setminus J) < x_i + l_i/8\}; \quad N^{22} := N^2 \setminus N^{21} . \quad (3.8)$$

Lemma 3.17. *If $h_i \leq l_i$ for $i \in N^2$ we have $v_i(M \setminus J) \geq x_i + \frac{l_i}{8}$.*

Proof. The value of any bag in H_i is less than $\frac{9}{8}$ for i (Lemma 3.4). By (3.3), the value of any bag which is not in $H_i \cup L_i$ is at most 1 and the total value of bags in L_i is at most

$$\frac{3}{4}l_i - x_i .$$

The total value of the items in M , in a normalized instance, is n so we have

$$\begin{aligned} v_i(M \setminus J) &\geq n - \left[\left(\frac{9}{8} h_i \right) + (n - h_i - l_i) + \left(\frac{3}{4} l_i - x_i \right) \right] \\ &= -\frac{h_i}{8} + \frac{l_i}{4} + x_i \geq x_i + \frac{l_i}{8}. \end{aligned}$$

Lemma 3.17 shows that (3.5) holds for all $i \in N^{22}$. Hence, by Lemma 3.5, if N^{21} is empty, then we do not need to update the MMS upper bounds, and we can proceed to the next stage of bag-filling. Note that Theorem 3.15 is still applicable for all agents in N^2 , which implies the following corollary.

Corollary 3.18. *For any agent $i \in N^{21}$, $\mu_i < 1$.*

The next theorem gives a new upper bound for these agents.

Theorem 3.19. *For any $i \in N^{21}$, $\mu_i \leq \alpha^* = \max\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$, where*

$$\begin{aligned} \alpha_1 &= \frac{4}{3} v_{i1} & \alpha_2 &= \frac{4}{3} (v_{in} + v_{i(n+1)}) \\ \alpha_3 &= \frac{4}{3} (v_{i(2n-1)} + v_{i(2n)} + v_{i(2n+1)}) & \alpha_4 &= \frac{4}{3} (v_{ik} + v_{ik'}) \\ \alpha_5 &= \max \left\{ \alpha : \frac{v_i(M \setminus J)}{\alpha} \geq \sum_{B_k : \frac{v_i(B_k)}{\alpha} < \frac{3}{4}} \left(\frac{3}{4} - \frac{v_i(B_k)}{\alpha} \right) + \frac{1}{8} \left| \left\{ B_k : \frac{v_i(B_k)}{\alpha} < \frac{3}{4} \right\} \right| \right\}, \end{aligned}$$

where k and k' are respectively the maximum value items from J and $M \setminus J$ that were not tentatively assigned, and $\langle N, M, V \rangle$ refers to the instance after we undo the tentative assignments in Line 6 of Algorithm 7.

Proof. For a contradiction, suppose $\mu_i = \beta$ and $\beta > \alpha^*$. We have

$$\begin{aligned} \beta > \alpha_1 &\implies v_{i1} < \frac{3}{4} \beta \\ \beta > \alpha_2 &\implies (v_{in} + v_{i(n+1)}) < \frac{3}{4} \beta \\ \beta > \alpha_3 &\implies (v_{i(2n-1)} + v_{i(2n)} + v_{i(2n+1)}) < \frac{3}{4} \beta \\ \beta > \alpha_4 &\implies (v_{ik} + v_{ik'}) < \frac{3}{4} \beta. \end{aligned}$$

From this, we can conclude that $v_i(S) < (\frac{3}{4})\mu_i$ for $S \in \{S_1, S_2, S_3\}$. Therefore, agent i will not be satisfied in Fixed-Assignment step. Further, since each bundle in $\{S_1, S_2, S_3\}$ is worth strictly less than $(\frac{3}{4})\mu_i$ to agent i , all S_4 bundles, which are tentatively assigned, (see Algorithm 9) have value at most μ_i , and hence all tentative assignments are valid reductions (Lemma 5.9). Also, since $\beta > \alpha_4$, no new pair that form S_4 will satisfy i . Furthermore, by normalizing v_{ij} 's (i.e., scaling them by $1/\beta$), the value of the bags in (3.2) will increase and consequently h_i will not decrease therefore, h_i will remain greater than 0. Consequently, since $h_i > 0$ and $\beta > \alpha_5$, agent i will remain in N^{21} even after we normalize her valuation with respect to β . This is because, by definition of α_5 , this is the maximum value that we can normalize the valuation such that agent i will not satisfy the definition of N^{21} in (3.8). Note that by normalizing the valuations, x_i and l_i will be affected that we have taken into account in definition of α_5 .

It implies that $\mu_i < \beta$ (Corollary 3.18), which is a contradiction. $\square$

Remark 3.20. *It is easy to see from (3.8) that if we scale the valuations by $\frac{1}{\alpha}$ for $\alpha = v(M \setminus J)/(x_i + \frac{l_i}{8})$, then agent i will not remain in N^{21} . However, α_5 can be more than α because if we scale the valuations by $\frac{1}{\alpha}$ for $0 < \alpha < 1$ both x_i and l_i will decrease (see (3.3)).*

Remark 3.21. α_5 can be computed in $O(n)$ time using a simple procedure. Sort the value of bags in L_i . Let u^k be the value of the k^{th} highest value bag in L_i . If $\alpha \in (1, u^1/(\frac{3}{4}))$ and we scale the valuations by $\frac{1}{\alpha}$ then all bags, which were in L_i will remain in L_i . Similarly, if $\alpha \in [u^k/(\frac{3}{4}), u^{k+1}/(\frac{3}{4}))$ then l_i will be reduced by k . After figuring out which bags remain in L_i and which bags do not, we can compute x_i with respect to α . Then, we can check whether there exists an α in the range for each k , starting with $k = 1$, that holds the definition of α_5 . If not, then we move to the next k .

Example 3.22. Consider an example where, after the execution of Algorithm 9, $n = 30$ (hence $|J| = 60$). The valuation of some agent $i \in N^{21}$ for items in J is shown in Table 3.3 and her valuation for bags is shown in Table 3.4. In this example, we show how to find α_5 .

Table 3.3: Valuation of agent i for items in J in Example 3.22

Item	1 – 27	28 – 30	31	32	33 – 60
v_{ij}	$\frac{29}{40}$	$\frac{15}{40}$	$\frac{14}{40}$	$\frac{13}{40}$	$\frac{12}{40}$

Table 3.4: The value of the bags in Example 3.22.

Bag	$B_1 - B_{27}$	B_{28}	B_{29}	B_{30}
$v_i(B_k)$	$\frac{41}{40}$	$\frac{27}{40}$	$\frac{28}{40}$	$\frac{29}{40}$
Bag type	H_i	L_i	L_i	L_i

Let $v_i(M \setminus J) = \frac{9}{40}$. First, observe that $h_i = 27, l_i = 3, x_i = \frac{6}{40}$, and $v_i(M \setminus J) = \frac{9}{40} < \frac{21}{40} = \frac{l_i}{8} + x_i$. It shows that agent i belongs to N^{21} .

In order to obtain α_5 , we check how the value of l_i would change after scaling the valuations of agent i by $\frac{1}{\alpha_5}$.

- Let $\alpha_5 > \frac{29}{40} \cdot \frac{4}{3}$ then $l_i = 3$. We check if there exists an α_5 in this range for which the following bound holds.

$$\frac{1}{\alpha_5} \times v_i(M \setminus J) \geq \frac{3}{8} + \left(\frac{3}{4} - \frac{v_i(B_{28})}{\alpha_5}\right) + \left(\frac{3}{4} - \frac{v_i(B_{29})}{\alpha_5}\right) + \left(\frac{3}{4} - \frac{v_i(B_{30})}{\alpha_5}\right) \quad (3.9)$$

- If not, let $\frac{28}{40} \cdot \frac{4}{3} < \alpha_5 \leq \frac{29}{40} \cdot \frac{4}{3}$ then $l_i = 2$ (B_{30} will not belong to L_i anymore). We check if there exists an α_5 in this range for which the following bound holds.

$$\frac{1}{\alpha_5} \times v_i(M \setminus J) \geq \frac{2}{8} + \left(\frac{3}{4} - \frac{v_i(B_{28})}{\alpha_5}\right) + \left(\frac{3}{4} - \frac{v_i(B_{29})}{\alpha_5}\right) \quad (3.10)$$

- If not, $\frac{27}{40} \cdot \frac{4}{3} < \alpha_5 \leq \frac{28}{40} \cdot \frac{4}{3}$ then $l_i = 1$. We check if there exists an α_5 in this range for which the following bound holds

$$\frac{1}{\alpha_5} \times v_i(M \setminus J) \geq \frac{1}{8} + \left(\frac{3}{4} - \frac{v_i(B_{28})}{\alpha_5}\right) \quad (3.11)$$

- Else, $\alpha_5 = \frac{27}{40} \cdot \frac{4}{3}$ ($l_i = 0$).

Note that in (3.9), (3.10), and (3.11) we have obtained the sum of $\frac{l_i}{8}$ and the new x_i after scaling the valuations. It takes at most l_i steps (3 steps in this example) to find the maximum α_5 for which one of the bounds above holds.

We pick an agent i with the lowest index in N^{21} and update i 's valuation as $v_{ij} \leftarrow \frac{v_{ij}}{\alpha}$, $\forall j \in M$ and repeat. In Theorem 3.23, we show that the number of updates is at most n^3 in the entire run of algorithm.

3.3.3 Bag-Filling and Running Time

Note that we reach the bag-filling stage only when N^{21} (see (3.8)) is empty. Here, we use Algorithm 6 of Section 3.2.2. Since N^{21} is empty, Lemma 3.5 is applicable, which implies a $\frac{3}{4}$ -MMS allocation. Next, we show that the entire algorithm runs in strongly polynomial time.

Theorem 3.23. *The entire algorithm runs in $O(n^5m)$ time for an ordered instance, and in $O(nm(n^4 + \log m))$ time for any instance.*

Proof. In Algorithm 7, normalization takes $O(mn)$ arithmetic operations. Both Fixed-Assignment and Tentative-Assignment procedures take at most $O(n^2m)$ arithmetic operations. The bag-filling procedure takes at most $O(n^2m)$ time. Each iteration of the while loop takes $O(n^2)$ arithmetic operations to find the set N^{21} of agents, $O(m)$ arithmetic operations to update valuations of an agent $i \in N^{21}$, and $O(n^2m)$ arithmetic operations to run Fixed-Assignment and Tentative-Assignment procedures. Therefore, each iteration of the while loop takes at most $O(n^2m)$ arithmetic operations.

To bound the number of iterations of the while loop, we upper bound the number of times it is run for a particular agent, say a , in N^{21} . Consider the first iteration when a is the lowest index agent, then if we update a 's valuation due to $\alpha \in \{\alpha_1, \alpha_2, \alpha_3\}$ (see Section 3.3.2), then a gets a fixed assignment after this iteration, and we will not see her again. If we update a 's valuation due to $\alpha = \alpha_4$, then in the future iterations, agent a will not be in N^{21} unless the instance is reduced due to a fixed assignment. This could happen only when another agent in N^{21} ends her iteration due to $\alpha \in \{\alpha_1, \alpha_2, \alpha_3\}$. Clearly, this can occur at most $O(n)$ time. Finally, if we update a 's valuation due to $\alpha = \alpha_5$, then agent a will not be in N^{21} again unless the instance is reduced due to either fixed or tentative assignment, which can affect the set of agents in N^{21} . This could happen only when another agent of N^{21} ends her iteration due to $\alpha \in \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$. Clearly, this can occur at most $O(n^2)$ time. Therefore, the maximum number of iterations of the while loop is at most $O(n^3)$. Since each iteration takes at most $O(n^2m)$ arithmetic operations, Algorithm 7 takes $O(n^5m)$ time. Note that this is for an ordered instance.

The reduction from the general instance to ordered instance is given in Chapter 2 Section 2.3.3. Algorithm 1 takes $O(mn \log m)$ time to make the original instance ordered by creating a sorted list of items for each agent. Next, we show that Algorithm 2 can be implemented in $O(mn)$ time. Initialize a binary array A of size m to all ones. $A[j]$ indicates whether item j is available or not; 1 means available. Each agent has a sorted list of items from the Algorithm 1. From the solution of the ordered instance using Algorithm 7, we can create an array B of size m , where $B[j]$ stores the agent who is assigned the item j . Observe that B can be constructed in $O(m)$ time. Then, for each item j from 1 to m , we get the agent $B[j]$ who is assigned j , and then we try to give $B[j]$ the highest item in her list. If her highest item is not available, which we can check from A , then we delete this item from $B[j]$'s list, and move down to the next highest item and repeat until we reach at an available item. We give this item to $B[j]$ and mark it assigned in A . Since the size of each agent's list is m , the total time is $O(mn)$. Using Algorithms 1 and 2 together with Algorithm 7, the running time of the entire procedure is $O(nm(n^4 + \log m))$. $\square$

3.4 Existence of $(\frac{3}{4} + \frac{1}{12n})$ -MMS Allocation

In this section, we show that our approach in Section 3.2 can be extended to obtain the existence of a $(\frac{3}{4} + \gamma)$ -MMS allocation for any given instance $\mathcal{I} = \langle N, M, V \rangle$ where $\gamma = \frac{1}{12n}$. We note that γ is a constant for the given instance, where $n := |N|$. We assume that the MMS value μ_i of each agent i is given. Finding

an exact μ_i is an NP-Hard problem, however a PTAS exists [48]. This implies a PTAS to compute a $(\frac{3}{4} + \gamma - \epsilon)$ -MMS allocation for any $\epsilon > 0$. Using the properties shown in Section 2.3, we normalize valuations so that $\mu_1 = 1, \forall i$ and assume that $\mathcal{I}$ is an ordered instance, i.e., $v_{i1} \geq \dots \geq v_{i|M|}, \forall i$. Our proof is algorithmic. If more than one agent satisfies the conditions (2.1) of valid reduction, then we choose one arbitrarily.

For the ease of exposition, we abuse notation and use M and N to denote the set of unallocated items and the set of agents who have not received any bundle yet, respectively. Further, we use j to denote the j^{th} highest value item in M . Moreover, we use $n := |N|$ and $m := |M|$. This is the reason we use γ , which is a constant for a given instance, to denote the approximation factor.

The approach is identical to Section 3.2. Here, we run Algorithm 4 with $\alpha = \frac{3}{4} + \gamma$. The analysis is also almost same. Here, we crucially use the extra $\frac{1}{8}$ (see (3.5) and the paragraph after it) to obtain a better factor. To avoid repetition, we only highlight the differences.

3.4.1 Initial Assignment

In Algorithm 5, we keep assigning a bundle $S \in \{S_1, S_2, S_3, S_4\}$ to agent i , if any, for which $v_i(S) \geq \frac{3}{4} + \gamma$. Then, we update M and N to respectively reflect the current unallocated items and agents who are not assigned any bundle yet. Observe that Lemma 5.9 is applicable here to show that assigning a bundle $S \in \{S_1, S_2, S_3\}$ is a *valid reduction*. However, it only applies for S_4 when $v_i(S_4) \leq 1, \forall i$.

As in Section 3.2, let $J_1 := \{1, \dots, n\}$ denote the set of first n items. Similarly, define $J_2 := \{n+1, \dots, 2n\}$ and $J := J_1 \cup J_2$. The following corollary is straightforward.

Corollary 3.24. *If $v_i(S) < \frac{3}{4} + \gamma, \forall i$ and $\forall S \in \{S_1, S_2, S_3\}$, then (i) $v_{ij} < \frac{3}{4} + \gamma, \forall j \in J_1$, (ii) $v_{ij} < \frac{3}{8} + \frac{\gamma}{2}, \forall j \in J_2$, and $v_{in} < \frac{3}{4} + \gamma - v_{i(n+1)}$, and (iii) $v_{ij} < \frac{1}{4} + \frac{\gamma}{3}, \forall j \in M \setminus J$, for all i .*

However, the value of S_4 might be strictly greater than 1 for an agent i . Corollary 3.24 implies that, $v_i(S_4) < \frac{3}{4} + \gamma + \frac{1}{4} + \frac{\gamma}{3} = 1 + \frac{4\gamma}{3}$. We use dummy items to fix the issue of extra $\frac{4\gamma}{3}$ lost in assigning S_4 .

Dummy items (D_1): For each removed S_4 in Algorithm 5, we add one dummy item d_j such that

$$v_i(\{d_j\}) = \frac{4\gamma}{3} \quad \forall i \in N. \quad (3.12)$$

Let D_1 denote the set of all dummy items after the termination of Algorithm 5. We note that dummy items will not be assigned to any agent. They are defined to make proofs easier. Later, we introduce two more sets D_2 and D_3 of dummy items in Section 3.4.2. Let $D := D_1 \cup D_2 \cup D_3$ with $D_2 = D_3 = \emptyset$ currently. The proof of the following corollary easily follows from Lemma 5.9.

Corollary 3.25. *For any remaining agent $i \in N$, $\mu_i^n(M \cup D) \geq 1$.*

The proof of the following lemma easily follows using (3.12) and Corollary 3.25.

Lemma 3.26. *When Algorithm 5 terminates, we have $\mu_i^n(M) \geq 1 - \frac{4\gamma r}{3}$ and $v_i(M) \geq n - \frac{4\gamma r}{3}, \forall i$, where r is the total number of rounds in Algorithm 5 when S_4 is assigned.*

3.4.2 Bag-Filling

The overall approach is same as in Section 3.2.2. We use the bag-filling procedure given in Algorithm 6 to satisfy the remaining agent by setting $\alpha = \frac{3}{4} + \gamma$. We initialize the bags with items in J as in (3.2) (see

Figure 3.1). Algorithm 6 has n rounds. In each round k , it starts a new bundle T with $T \leftarrow B_k$. If there is an agent who values T to be at least $\frac{3}{4} + \gamma$, then assign T to such an agent. Otherwise, keep adding items from $M \setminus J$ to T one by one until someone values T at least $\frac{3}{4} + \gamma$.

For correctness, we need to show that there are enough items in $M \setminus J$ to add on top of each bag in (3.2) so that each agent gets a bundle that they value at least $\frac{3}{4} + \gamma$. For this, we first divide agents into two types: $N^1 := \{i \in N \mid v_i(B_k) \leq 1 + \frac{3\gamma}{2}, \forall k\}$ and $N^2 := N \setminus N^1$. Next, we update the notations of (3.3) to reflect the improved bound. For an agent $i \in N^2$, define

$$\begin{aligned} L_i &:= \{B_k : v_i(B_k) < \frac{3}{4} + \gamma\}; & l_i &:= |L_i| \\ H_i &:= \{B_k : v_i(B_k) > 1 + \frac{3\gamma}{2}\}; & h_i &:= |H_i| \\ x_i &:= (\frac{3}{4} + \gamma)l_i - \sum_{k: B_k \in L_i} v_i(B_k). \end{aligned} \tag{3.13}$$

The proof of the following lemma is an easy extension of Lemma 3.4, hence omitted.

Lemma 3.27. *For an agent $i \in N^2$, (i) $l_i > 0$ and $h_i > 0$, (ii) $v_{i1} > \frac{5}{8} + \gamma$, (iii) $v_i(B_k) < \frac{9}{8} + \frac{3\gamma}{2}, \forall k$, and (iv) $v_{ij} < \frac{1}{8}, \forall j \in M \setminus J$.*

The proof of the following lemma is an extension of the proof of Lemma 3.5, and is in Section 3.6.

Lemma 3.28. *If $v_i(M \setminus J) \geq x_i + \frac{l_i}{8} - \frac{1}{8}, \forall i \in N^2$, then Algorithm 6 with $\alpha = \frac{3}{4} + \gamma$ gives every agent a bundle that they value at least $\frac{3}{4} + \gamma$.*

Now, we only need to show that for each $i \in N^2$, we have

$$v_i(M \setminus J) \geq x_i + \frac{l_i}{8} - \frac{1}{8} \tag{3.14}$$

We start with a few lemmas. Recall from (3.1) that $\mathcal{P}_i^n(M)$ denote the set of partitions where the value of each bundle is at least 1. The proofs of the following two lemmas are extensions of the corresponding Lemmas 3.6 and 3.8 respectively, and are given in Section 3.6.

Lemma 3.29. *For an agent $i \in N^2$, there exists a bundle P_k in every partition $P = \{P_1, \dots, P_n\} \in \mathcal{P}_i^n(M \cup D)$ such that $v_i(P_k \setminus J) > \frac{1}{4} - \gamma$.*

Lemma 3.30. *For every agent i , there exists a partition $P = \{P_1, \dots, P_n\} \in \mathcal{P}_i^n(M \cup D)$ such that each $P_k \in P$ has at most one item j with $v_{ij} > \frac{5}{8} + \gamma$.*

Fix an agent, say a , in N^2 . For simplicity, until the end of this section, when we use *value* of an item or a bundle, we mean their value for agent a (unless mentioned otherwise). Recall that we need to show (3.14). Let $P = \{P_1, \dots, P_n\} \in \mathcal{P}_a^n(M \cup D)$ be a partition satisfying the condition of Lemma 3.30.

We manipulate P as follows: First, for each $B_i = \{j, j'\}$ with $\frac{3}{4} + \gamma \leq v_a(B_i) \leq 1 + \frac{3\gamma}{2}$, if $j \in P_k$ and $j' \in P_{k'}$, then we turn P_k and $P_{k'}$ into two new bundles $\{j, j'\}$ and $(P_k \cup P_{k'} \cup \{d_j\} \setminus \{j, j'\})$, where d_j is a new dummy item with $v_a(d_j) := \frac{3\gamma}{2}$.

Dummy items (D_2): Let D_2 denote the set of all dummy items added during the manipulation of P . Observe that $|D_2| = n - h_a - l_a$. Each $d_j \in D_2$ values $\frac{3\gamma}{2}$ for agent a . Therefore,

$$v_a(D_2) = \frac{3\gamma|D_2|}{2}. \tag{3.15}$$

Later we introduce one more set D_3 of dummy items. Recall that $D = D_1 \cup D_2 \cup D_3$ where D_1 is defined in (3.12) and $D_3 = \emptyset$ currently. Observe that $v_a(P_k \cup P_{k'} \cup \{d_j\} \setminus \{j, j'\}) \geq 1$. Hence, we can assume that for each $B_i \notin L_a \cup H_a$ (as defined in (3.13)), there exists a $P_k = B_i$ and all other bundles value at least 1. Second, we re-enumerate the bundles in P such that

- $P_1, \dots, P_{t_1}$: each has an item j of value more than $\frac{5}{8} + \gamma$ and $v_a(B_j) < \frac{3}{4} + \gamma$.
- $P_{t_1+1}, \dots, P_{t_2}$: each has an item j of value more than $\frac{5}{8} + \gamma$ and $v_a(B_j) > 1$ (Observe that there are h_a such bundles, so $t_2 - t_1 = h_a$).
- $P_{t_3+1}, \dots, P_n$: each such $P_k = B_{k'}$ for some k' and $\frac{3}{4} + \gamma \leq v_a(B_{k'}) \leq 1 + \frac{3\gamma}{2}$.
- $P_{t_2+1}, \dots, P_{t_3}$ be the remaining bundles (Observe that $t_3 = l_a + h_a$).

Definition 3.31. Define $y := v_{ak'}$ where k' is the highest value item in the set $\{j \in J_2 \cap B_k, B_k \in H_a\}$.

Observe that $\frac{1}{4} + \frac{\gamma}{2} < y < \frac{3}{8} + \frac{\gamma}{2}$. For instance, in Figure 3.1, if no bag from $\{B_{k+1}, \dots, B_n\}$ is in H_a but $B_k \in H_a$, then $y = v_{a(2n-k+1)}$. This also implies that $v_{a1} \geq \dots \geq v_{ak} > \frac{5}{8} + \gamma$. Moreover, $y = v_{a(2n-k+1)} \leq v_{a(2n-k)} \leq \dots \leq v_{a(k+1)}$. The proof of the following lemma is an easy extension of the proof of Lemma 3.12, and hence omitted.

Lemma 3.32. For any bundle $B_k = \{j, j'\} \in L_a$ such that $v_{aj}, v_{aj'} < \frac{5}{8} + \gamma$, we have $v_{aj}, v_{aj'} \geq y$.

Next, we show the extension of Lemma 3.13, whose proof is given in Section 3.6.

Lemma 3.33. For agent $a \in N^2$, if $t_1 = 0$ then we have

1. $x_a \leq (\frac{3}{4} + \gamma - 2y)l_a$
2. $v_a((M \cup D) \setminus J) \geq \max\{x_a + l_a(\frac{1}{4} - \gamma) - yz + z(\frac{1}{4} - \gamma), z(\frac{1}{4} - \gamma)\}$, where $z = \max\{2h_a - |\bigcup_{t=t_1+1}^{t_2} P_t \cap J|, 0\}$.

The following lemma is an extension of Theorem 3.14, and its proof is given in Section 3.6.

Theorem 3.34. For agent $a \in N^2$, if $t_1 = 0$ then

$$v_a(M \cup D \setminus J) \geq x_a + \frac{l_a}{8}. \quad (3.16)$$

Next, we handle the general case when $t_1 > 0$. Let $L_a^1 \subseteq L_a$ be the set of bags in L_a , which have one item with value more than $\frac{5}{8} + \gamma$, and $L_a^2 = L_a \setminus L_a^1$. Clearly, $|L_a^1| = t_1$ and $|L_a^2| = l_a - t_1$. Note that $|L_a^2| \geq 1$ because $B_n \in L_a^2$.

Dummy items (D_3): For each bag in L_a^1 we add a dummy item d_j where $v_a(d_j) = \gamma$. Therefore,

$$v_a(D_3) = |D_3| \cdot \gamma = |L_a^1| \cdot \gamma < l_a \cdot \gamma. \quad (3.17)$$

The following theorem is an extension of Theorem 3.15, and its proof is given in Section 3.6.

Theorem 3.35. For agent $a \in N^2$, we have $v_a((M \cup D) \setminus J) \geq x_a + \frac{l_a}{8}$.

Next, we obtain the bound without the dummy items in the next theorem.

Theorem 3.36. For any agent $a \in N^2$, $v_a(M \setminus J) \geq x_a + \frac{l_a}{8} - \frac{1}{8}$.

Proof. From Lemma 3.26, (3.15), and (3.17), the total value of the items in D is at most

$$v_a(D) \leq \frac{4\gamma r}{3} + \frac{3\gamma|D_2|}{2} + \gamma l_a \leq \frac{3\gamma n}{2} \leq \frac{1}{8},$$

because $|D_2| + r \leq (n - h_a - l_a)$ and $\gamma = \frac{1}{12n}$, where n is the number of agents in the original instance. This, together with Theorem 3.35, proves the theorem. $\square$

3.4.3 Better Approximation Factor?

The algorithm is specifically designed for the approximation factor of $\frac{3}{4}$. The first issue that the algorithm faces by increasing the approximation factor, is that removing S_4 will no longer be a valid reduction (see Lemma 5.9 and Section 3.4.1). However, as we showed in Section 3.4, with the use of the extra $\frac{1}{8}$ value in $M \setminus J$ (Lemma 3.5 and Theorem 3.15), we could use the same algorithm for the factor of $\frac{3}{4} + \gamma$ with $\gamma = \frac{1}{12n}$. This raises a natural question of whether this is the best algorithm can do, i.e., are there tight examples for which the same algorithm cannot guarantee better than $\frac{3}{4} + \frac{1}{12n}$? We leave this as an interesting open question. However, we can show that our algorithm does not guarantee a factor greater than $\frac{3}{4} + \frac{1}{12}$.

Example 3.37. Let $\mathcal{I} = \langle N, M, V \rangle$ be an instance for which we want $\alpha(= \frac{5}{6} + \epsilon)$ -MMS allocation for some $\epsilon > 0$. Let $N = \{1, 2\}$ be the set of agents with identical valuations, $M = \{1, 2, \dots, m\}$ be the set of items where $m = \lceil \frac{1}{3\epsilon} \rceil + 4$, and $V = v$. Agents' valuation for each item is shown in Table 3.5.

Table 3.5: Valuation of agents for item $j, j \in M$ in Example 3.37

Item	1	2 - 4	5 - m
v_j	$\frac{5}{6}$	$\frac{1}{3}$	$\frac{\epsilon}{2}$

Clearly, the MMS partition is $P_1 = \{2, 3, 4\}$ and $P_2 = \{1\} \cup \{5, \dots, m\}$ of M , which implies $\mu_i^n(M) = 1, \forall i$. When we run Algorithm 4 on $\mathcal{I} = \langle N, M, V \rangle$ then there will not be any initial assignment, and it will run the bag-filling procedure with bags $B_1 = \{1, 4\}$ and $B_2 = \{2, 3\}$, which will not output an α -MMS allocation.

3.5 Conclusions

We developed a new approach that gives a simple algorithm for showing the existence of a $\frac{3}{4}$ -MMS allocation. Furthermore, we showed that our approach is powerful enough to be easily extended to obtain (i) a strongly polynomial time algorithm to find a $\frac{3}{4}$ -MMS allocation, and (ii) the existence of a $(\frac{3}{4} + \frac{1}{12n})$ -MMS allocation, improving the best previous factor. Consequently, this gives a PTAS for finding a $(\frac{3}{4} + \frac{1}{12n} - \epsilon)$ -MMS allocation for any $\epsilon > 0$. An interesting question is to find the maximum γ for which the proposed algorithm obtains $(\frac{3}{4} + \gamma)$ -MMS allocation. We could show that our algorithm fails for $\gamma > \frac{1}{12}$, but the question remains open for $\gamma \in (\frac{1}{12n}, \frac{1}{12}]$.

Further, it could be worth exploring whether extending the approach yields a $\frac{4}{5}$ -MMS allocation. Such an extension would be challenging because after the initial greedy assignments, there will be $3n$ high-value items, and this would make the process of initializing the bag-filling procedure harder due to too many items to handle and also the value of some bags might exceed significantly more than 1.

3.6 Missing Proofs

Lemma 3.28. *If $v_i(M \setminus J) \geq x_i + \frac{l_i}{8} - \frac{1}{8}, \forall i \in N^2$, then Algorithm 6 with $\alpha = \frac{3}{4} + \gamma$ gives every agent a bundle that they value at least $\frac{3}{4} + \gamma$.*

Proof. This is proof by contradiction. Suppose the algorithm stops at round t because there are not enough items in $L (= M \setminus J)$ to satisfy any remaining agent i , i.e., $v_i(B_t \cup L) < \frac{3}{4} + \gamma$.

If $i \in N^1$, each removed bundle in rounds $k \in [t-1]$, has value of at most $1 + \frac{3\gamma}{2}$ for agent i . Because, if $v_i(B_k) \geq \frac{3}{4} + \gamma$ for $k \in [t-1]$, no more item has been added to $T = B_k$. Also, if $v_i(B_k) < \frac{3}{4} + \gamma$ for $k \in [t-1]$ before adding the last item (if any) to T the value of T is less than $\frac{3}{4} + \gamma$ and from Corollary 3.24, $v_{ij} < \frac{1}{4} + \frac{\gamma}{3}$ for $j \in L$. Therefore, at the end of the round $v_i(T) < 1 + \frac{3\gamma}{2}$.

This implies that the total value of assigned bundles in round 1 to $t-1$ is at most $(t-1)(1 + \frac{3\gamma}{2})$. Let n' be the number of agents remaining after Algorithm 5, and n be the number of agents in the original instance. We have $v_i(M) \geq n' - \frac{4\gamma r}{3}$ due to Lemma 3.26. Therefore, the value of items in L before the round t starts is at least

$$v_i(L) \geq n' - \frac{4\gamma r}{3} - ((t-1)(1 + \frac{3\gamma}{2}) + v_i(B_t) + (n' - t)(1 + \frac{3\gamma}{2})).$$

This implies that

$$v_i(L) + v_i(B_t) \geq 1 - \frac{3(r + (n' - 1))\gamma}{2} \geq 1 - \frac{3(n-1)\gamma}{2} \geq \frac{7}{8} + \frac{1}{8n} \geq \frac{3}{4} + \gamma,$$

where we use $\gamma = \frac{1}{12n}$, which is a contradiction.

If $i \in N^2$, then since at round t , $v_i(T) < \frac{3}{4} + \gamma$, we have $B_t \in L_i$. Consider a round $k \in [t-1]$. If $B_k \notin L_i$, then $T = B_k$ has been assigned to someone with no additional items added to T from L because $i \in \Gamma(T)$. If $B_k \in L_i$, then in round k , before adding the last item (if any) to T , the value of i for T is less than $\frac{3}{4} + \gamma$ and from Lemma 3.27, all items in L have value of at most $\frac{1}{8}$. Therefore, if $B_k \in L_i$, the value of the assigned bag for i in round k is less than $\frac{7}{8} + \gamma$. Hence, in the beginning of the round t ,

$$v_i(L) \geq (x_i + \frac{l_i}{8} - \frac{1}{8}) - (x_i - (\frac{3}{4} + \gamma - v_i(B_t)) + (l_i - 1)/8) = \frac{3}{4} + \gamma - v_i(B_t),$$

which is a contradiction. □

Lemma 3.29. *For an agent $i \in N^2$, there exists a bundle P_k in every partition $P = \{P_1, \dots, P_n\} \in \mathcal{P}_i^n(M \cup D)$ such that $v_i(P_k \setminus J) > \frac{1}{4} - \gamma$.*

Proof. If there exists a bundle P_k with exactly one item j from J , then $v_i(P_k \setminus J) = 1 - v_{ij} > \frac{1}{4} - \gamma$ because value of every item is less than $\frac{3}{4} + \gamma$. Otherwise, each bundle has exactly two items from J , which implies that one of the bundles, say P_k , has two items j_1, j_2 from the set $\{n\} \cup J_2$. Since $v_{ij_1} + v_{ij_2} \leq v_{in} + v_{i(n+1)} < \frac{3}{4} + \gamma$, $v_i(P_k \setminus J) > \frac{1}{4} - \gamma$. □

Lemma 3.30. *For every agent i , there exists a partition $P = \{P_1, \dots, P_n\} \in \mathcal{P}_i^n(M \cup D)$ such that each $P_k \in P$ has at most one item j with $v_{ij} > \frac{5}{8} + \gamma$.*

Proof. We show that if there are two items j_1, j_2 with each value more than $\frac{5}{8} + \gamma$ for an agent i in a bundle P_k of $P \in \mathcal{P}_i^n(M \cup D)$, then we can construct another $P' \in \mathcal{P}_i^n(M \cup D)$ where this is not true. Corollary 3.24 implies that there must exist another bundle $P_{k'}$ for which $\max_{j \in P_{k'}} v_{ij} < \frac{3}{8} + \frac{\gamma}{2}$. If there exists a partition $Q_{k'}^1$ and $Q_{k'}^2$ of items in $P_{k'}$ such that value of each $Q_{k'}^1$ and $Q_{k'}^2$ is at least $\frac{3}{8} - \gamma$, then we can rearrange

items in $P_k \cup P_{k'}$ and make two new bundles $(\{j_1\} \cup Q_{k'}^1)$ and $(\{j_2\} \cup Q_{k'}^2)$. Clearly, the value of each bundle is at least 1 and each has exactly one item with value more than $\frac{5}{8} + \gamma$.

If no such $Q_{k'}^1$ and $Q_{k'}^2$ exists, then we claim that there exists a partition of $P_{k'}$ into three sets with each value less than $\frac{3}{8}$. We can find this partition as follows: Initialize three empty bundles, and repeatedly add the highest value item of $P_{k'}$ to the bundle with the lowest value. For a contradiction, suppose one of the three bundles has value more than $\frac{3}{8}$, then the sum of the values of the other two sets must be less than $\frac{3}{8}$ because otherwise, they make a partition of two with each value more than $\frac{3}{8} - \gamma$. This means that at least one of the bundles must have a value less than $\frac{3}{16}$. This implies that the value of the largest bag before adding the last item must be less than $\frac{3}{16}$ and the last item also should value less than $\frac{3}{16}$, which is a contradiction. Therefore, there exists a partition $Q_{k'}^1$, $Q_{k'}^2$ and $Q_{k'}^3$ of $P_{k'}$ such that each value less than $\frac{3}{8}$.

According to Lemma 3.29, there exists a bundle $P_{\hat{k}} \in \mathcal{P}_i^n(M)$ such that $v_i(P_{\hat{k}} \setminus J) > \frac{1}{4} - \gamma$. Let $v_i(P_{\hat{k}}) = 1 + \delta$ for some $\delta \geq 0$. Observe that $P_{k'}$ cannot be same as $P_{\hat{k}}$ and P_k , otherwise we would have made two bundles earlier, each with value at least 1 and exactly one item more than $\frac{5}{8} + \gamma$. We initialize three bags:

$$\text{bag 1 : } \{j_1\} \cup Q_{k'}^1 \quad \text{bag 2 : } \{j_2\} \cup Q_{k'}^2 \quad \text{bag 3 : } (P_k \setminus \{j_1, j_2\}) \cup (P_{\hat{k}} \cap J) \cup Q_{k'}^3$$

Observe that the value of each of bag 1 and bag 2 is at most $\frac{9}{8} + \frac{3\gamma}{2}$, and the total value of all items in P_k , $P_{k'}$ and $P_{\hat{k}}$ is at least $3 + \frac{2}{8} + \delta + 2\gamma$. We sort the remaining items in decreasing order and add them one by one to a bag with the lowest value. Since the value of the last item added to each bag has a value of at most $\frac{1}{8}$ (from Lemma 3.27), each bag has a value of at least 1 and it has at most one item with value more than $\frac{5}{8} + \gamma$. We repeat this process for all bundles with two item of value more than $\frac{5}{8} + \gamma$ to find a desired partition. $\square$

Lemma 3.33. *For agent $a \in N^2$, if $t_1 = 0$ then we have*

1. $x_a \leq (\frac{3}{4} + \gamma - 2y)l_a$
2. $v_a((M \cup D) \setminus J) \geq \max\{x_a + l_a(\frac{1}{4} - \gamma) - yz + z(\frac{1}{4} - \gamma), z(\frac{1}{4} - \gamma)\}$, where $z = \max\{2h_a - |\bigcup_{t=t_1+1}^{t_2} P_t \cap J|, 0\}$.

Proof. For the first part, since $t_1 = 0$, we have $v_a(B_k) \geq 2y, \forall B_k \in L_a$ (Lemma 3.32). This implies that $x_a = (\frac{3}{4} + \gamma)l_a - \sum_{k: B_k \in L_a} B_k \leq (\frac{3}{4} + \gamma - 2y)l_a$.

For the second part, in $\bigcup_{k: B_k \in H_a} B_k$ there are exactly $2h_a$ items from J . If there are at least $2h_a$ items in $\bigcup_{t=t_1+1}^{t_2} P_t \cap J$ then $z = 0$. On the other hand, if there are $2h_a - z$ items in $\bigcup_{t=t_1+1}^{t_2} P_t \cap J$, then it means that there are at least z bundles in $P_{t_1+1}, \dots, P_{t_2}$ with exactly one item from J . Each of these bundles need more than $\frac{1}{4} - \gamma$ value of items from $M \setminus J$ to become 1.

Next, if $z = 0$, then all bundles in $\{P_{t_2+1}, \dots, P_{t_3}\}$ need at least $x_a + l_a(\frac{1}{4} - \gamma)$ value of items from $M \setminus J$ to become 1 because the value of each item in the bundles of L_a is at least y (follows from Definition 3.31 and the construction of B_k 's in (3.2)). If $z > 0$, then all bundles in $\{P_{t_2+1}, \dots, P_{t_3}\}$ need at least $x_a + l_a(\frac{1}{4} - \gamma) - yz$ value of items from $M \setminus J$ to become one because each of the z items has value at most y (Lemma 3.32). Therefore, in total, there should be at least $x_a + l_a(\frac{1}{4} - \gamma) - yz + z(\frac{1}{4} - \gamma)$ value of items from $M \setminus J$ in $\{P_{t_1+1}, \dots, P_{t_3}\}$.

Further, in case z is large such that the value of $x_a + l_a(\frac{1}{4} - \gamma) - yz + z(\frac{1}{4} - \gamma)$ is negative, we still need $z(\frac{1}{4} - \gamma)$ value of items from $M \setminus J$ to make all bundles in $\{P_{t_1+1}, \dots, P_{t_2}\}$ value 1. $\square$

Theorem 3.34. For agent $a \in N^2$, if $t_1 = 0$ then

$$v_a(M \cup D \setminus J) \geq x_a + \frac{l_a}{8} . \quad (3.16)$$

Proof. Using Lemma 3.33(2) there are two cases for the maximum of $x_a + l_a(\frac{1}{4} - \gamma) - yz + z(\frac{1}{4} - \gamma)$ and $z(\frac{1}{4} - \gamma)$. We prove the claim for each case separately.

Case 1: Suppose $x_a + l_a(\frac{1}{4} - \gamma) - yz + z(\frac{1}{4} - \gamma) \geq z(\frac{1}{4} - \gamma)$, then $z \leq \frac{x_a + l_a(\frac{1}{4} - \gamma)}{y}$. Further, Lemma 3.33(1) implies $y \leq \frac{l_a(\frac{3}{4} + \gamma) - x_a}{2l_a}$. Using these, we get

$$\begin{aligned} v_a(M \cup D \setminus J) &\geq x_a + l_a(\frac{1}{4} - \gamma) - yz + z(\frac{1}{4} - \gamma) \\ &\geq \frac{x_a(\frac{1}{4} - \gamma) + l_a(\frac{1}{4} - \gamma)^2}{y} \\ &\geq \frac{2x_al_a(\frac{1}{4} - \gamma) + 2l_a^2(\frac{1}{4} - \gamma)^2}{l_a(\frac{3}{4} + \gamma) - x_a} . \end{aligned}$$

We need to show that

$$\frac{2x_al_a(\frac{1}{4} - \gamma) + 2l_a^2(\frac{1}{4} - \gamma)^2}{l_a(\frac{3}{4} + \gamma) - x_a} - \left(x_a + \frac{l_a}{8}\right) \geq 0 . \quad (3.18)$$

For $n \leq 4$ (i.e., number of agents in the original instance is at most 4), there is a simpler way to prove this claim. However, there are better approximation factors available for them in any case [31], so we assume that $n > 4$. Since $\gamma = \frac{1}{12n}$, it decreases as n increases. Therefore, it is enough to show (3.18) for $n = 5$. For this, we put $\gamma = \frac{1}{60}$ in (3.18) and get,

$$\begin{aligned} \frac{2x_al_a(\frac{1}{4} - \frac{1}{60}) + 2l_a^2(\frac{1}{4} - \frac{1}{60})^2}{l_a(\frac{3}{4} + \frac{1}{60}) - x_a} - \left(x_a + \frac{l_a}{8}\right) &= \frac{7(30x_al_a + 7l_a^2)}{15(23l_a - 30x_a)} - \frac{8x_a + l_a}{8} \\ &= \frac{3600x_a^2 - 630x_al_a + 47l_a^2}{120(23l_a - 30x_a)} \\ &> \frac{36(10x_a - l_a)^2 + l_a(11l_a + 90x_a)}{120(23l_a - 30x_a)} \\ &\geq 0 . \end{aligned}$$

The last inequality follows because the denominator is positive using the definition of x_a .

Case 2: Suppose $x_a + l_a(\frac{1}{4} - \gamma) - yz + z(\frac{1}{4} - \gamma) \leq z(\frac{1}{4} - \gamma)$. Then, using Lemma 3.33(1), we have

$$v_a((M \cup D) \setminus J) \geq z(\frac{1}{4} - \gamma) \geq \frac{x_a(\frac{1}{4} - \gamma) + l_a(\frac{1}{4} - \gamma)^2}{y} \geq x_a + \frac{l_a}{8} ,$$

where the last inequality follows from the Case 1. □

Theorem 3.35. For agent $a \in N^2$, we have $v_a((M \cup D) \setminus J) \geq x_a + \frac{l_a}{8}$.

Proof. Let $J_{\min} = \bigcup_{B_k \in L_a^1} B_k \cap J_2$. Then, we have $v_{aj} < \frac{1}{8}, \forall j \in J_{\min}$ because these items are bagged with an item value more than $\frac{5}{8} + \gamma$ and together they have value less than $\frac{3}{4} + \gamma$. Therefore, $v_a(J_{\min}) < \frac{t_1}{8}$. Let

$x'_a := (\frac{3}{4} + \gamma)t_1 - \sum_{B_k \in L_a^1} v_a(B_k)$, and $x''_a := x_a - x'_a$. For agent a , we can treat items of $J_{\min}$ as low-value items so we will prove:

$$v_a((M \cup D) \setminus J) + v_a(J_{\min}) \geq x'_a + \frac{t_1}{4} + x''_a + (l_a - t_1)/8. \quad (3.19)$$

Since $v_a(J_{\min})$ is at most $\frac{t_1}{8}$, (3.19) directly implies the theorem. Let us call items in $(M \setminus J) \cup J_{\min} \cup D$ as *filler* items (each has value $< \frac{1}{8}$) and items in $J \setminus J_{\min}$ as *base* items (each has value $> \frac{1}{4} + \frac{\gamma}{3}$).

In re-enumerated $\mathcal{P}_a^n(M \cup D)$, recall that each of $\{P_1, \dots, P_{t_1}\}$ contains a base item whose value is at least the maximum of all the items in the remaining bundles. Further, if there are two base items in any bundle P_k in $\{P_1, \dots, P_{t_1}\}$, then $v_a(P_k) > 1$. If each bundle in $\{P_1, \dots, P_{t_1}\}$ has exactly one base item (t_1 bundles in total) then items of value at least $x'_a + t_1(\frac{1}{4} - \gamma)$ are needed to make all these bundles at least 1 and there are t_1 items in D_3 with value γ . This, together with Theorem 3.34, proves the bound. On the other hand, if there are more than t_1 *base* items in $\{P_1, \dots, P_{t_1}\}$, then for each extra base item j , there are two cases. If j comes from $B_k \in L_a^2$ then clearly, x'_a will decrease by less than v_{aj} but x''_a will increase by v_{aj} , so the bound only improves. For the other case, if j comes from $B_k \in H_a$ then it will make $B_k \setminus \{j\}$ to need more low-value items than the bundle in $\{P_1, \dots, P_{t_1}\}$ who gets j to become one, so x'_a will only increase and x''_a stays same, and hence the bound only improves. This completes the proof. $\square$

Chapter 4

4/7-GMMS Allocation After Donating Some Items

4.1 Introduction

Even though MMS provides a intuitive fairness guaranty in some instances an MMS allocation can be unsatisfactory. For this reason, Barman et al. [14] introduced the notion of Groupwise MMS (GMMS) allocations. Informally, an allocation is GMMS if every sub-allocation is an MMS allocation. GMMS is a generalization of MMS allocation and consequently, similar to MMS allocation, a GMMS allocation may not exist. In this chapter we define the notion of GMMS allocation and we propose an algorithm that obtains a $\frac{4}{7}$ -GMMS allocation after donating at most n items.

4.2 Problem Statement and Notations

Given $\mathcal{I} = \langle N, M, V \rangle$, the MMS value of the agents and a MMS allocation is defined in Definitions 2.1 and 2.2.

Definition 4.1 (GMMS Allocation). *An allocation $A = (A_1, \dots, A_n)$ is a GMMS allocation if*

$$v_a(A_a) \geq \mu_a^{|N'|} \left(\bigcup_{i \in N'} A_i \right) \quad \forall a \in N, N' \subseteq N : N' \ni i .$$

Given $\mathcal{I} = \langle N, M, V \rangle$, consider an MMS partition of agent $i \in N$ $P_i^n(M)$. From Definition 2.1, $P_i^n(M)$ and $\mu_i^n(M)$ we can conclude that there exists $j \in [n]$ for which we have $v_i(P_j) = \mu_i^n(M)$ (because otherwise, if $v_i(P_j) > \mu_i^n(M)$ for all $j \in [n]$, it contradicts the definition of $\mu_i^n(M)$).

The partition for $P_i^n(M)$ is not necessarily unique but we use the notation as one of such partitions.

4.3 Summary of the 1/2-GMMS Algorithm

In this chapter, we mainly continue the work from [14] which provides an algorithm to obtain $\frac{1}{2}$ -GMMS allocation. We alternate their $\frac{1}{2}$ -GMMS algorithm to get an algorithm which provides $\frac{4}{7}$ -GMMS allocation.

In the $\frac{1}{2}$ -GMMS algorithm (Algorithm 10), they make a directed graph $G = (N, E)$ which $(i, j) \in E$ if agent i envies agent j . Then, they start with an agent and add her favorite item to her bundle. Then they update G . If a cycle is created, they remove it by re-allocating the existing bundles. More precisely, assume cycle C is created after adding an item to a source agent. If we re-allocate bundles in a way that each agent in C gets the bundle of their next agent in the cycle, everyone will get a bundle which they value more than their previous bundle and the cycle C is removed. After removing all cycles, the algorithm picks another source agent it adds her favorite item among un-allocated items to her bundle. The algorithm continues until all items are allocated.

Algorithm 10: $\frac{1}{2}$ -GMMS Algorithm

Input: $I = \langle N, M, V \rangle$
Output: $A = (A_1, \dots, A_n)$ which is $\frac{1}{2}$ -GMMS for all $i \in N$

- 1 let $G = (N, E)$ be the envy graph
- 2 initialize $A_i \leftarrow \emptyset$ for all $i \in N$
- 3 **while** $|M| > 0$ **do**
- 4 choose a node i which is a source
- 5 Add i 's favorite j item in M to A_i
- 6 Remove j from M
- 7 Update envy graph G
- 8 **while** *there is cycle in G* **do**
- 9 remove a cycle by exchanging bundles

Lemma 4.2. *For a given allocation $A = (A_1, \dots, A_n)$ of items in M and a fixed agents $a \in N$, let N' be a subset of N where $a \in N'$ and let $\hat{N}' := \{i \in N' : |A_i| > 1\}$. If*

$$v_a(A_a) \geq \alpha \mu_a^{|\hat{N}'|} \left(\bigcup_{i \in \hat{N}'} A_i \right)$$

then

$$v_a(A_a) \geq \alpha \mu_a^{|N' \cup \hat{N}'|} \left(\bigcup_{i \in N' \cup \hat{N}'} A_i \right).$$

Proof. Adding k items and k agents to a set of items and agents will not increase the MMS value of the agent. Therefore, $\mu_a^{|\hat{N}'|} \left(\bigcup_{i \in \hat{N}'} A_i \right) \geq \mu_a^{|N'|} \left(\bigcup_{i \in N'} A_i \right)$. $\square$

Corollary 4.3. *An allocation $A = (A_1, \dots, A_n)$ is α -GMMS if for all $a \in N$ and $N' \in \{S \subseteq N : a \in S, |A_i| \geq 2 \ \forall i \neq a \in S\}$ we have*

$$v_a(A_a) \geq \alpha \mu_a^{|N'|} \left(\bigcup_{i \in N'} A_i \right).$$

Theorem 4.4. *Algorithm 10 gives a $\frac{1}{2}$ -GMMS allocation.*

Proof. Let $A = (A_1, \dots, A_n)$ be the output of Algorithm 10. Consider a fixed agent $a \in N$. Let $N' \ni a$ be a set of agents. Without loss of generality, assume that for all $i \neq a \in N'$ we have $|A_i| \geq 2$ (Corollary 4.3). Let $i \neq a$ be an agent in N' and let j_i be the last item which is added to A_i . Since the algorithm only adds items to the source agents we have

$$v_a(A_a) \geq v_a(A_i \setminus \{j_i\}).$$

Since agents pick their favorite available item at each round, we have

$$v_a(A_i) \geq v_a(j_i).$$

Because, otherwise, agent a would have chosen j_i instead of the item(s) in A_i . Therefore, we have

$$v_a(A_a) \geq \frac{1}{2}(v_a(A_i)).$$

This holds for any other agent in N' , i.e.,

$$v_a(A_a) \geq \frac{1}{2}(v_a(A_i)) \quad \forall i \neq a \in N'. \quad (4.1)$$

If we add all the inequalities in (4.1) along with the trivial inequality $v_a(A_a) \geq 1/2 v_a(A_a)$, we have

$$v_a(A_a) \geq \frac{1}{2} \frac{\sum_{i \in N'} v_a(A_i)}{|N'|} \geq \frac{1}{2} \mu_a^{|N'|} \left(\bigcup_{i \in N'} A_i \right).$$

□

4.4 4/7-GMMS Allocation

Before introducing the algorithm to get $\frac{4}{7}$ -GMMS, we explain the idea to get this allocation in the following lemma.

Lemma 4.5. *If for $a, i \in N$ with $i \neq a$ we have*

$$v_a(A_i) \geq \begin{cases} \max_{j \in A_i} v_{aj} & |A_i| = 2 \text{ (first condition)} \\ \frac{2}{3} v_a(A_i) & |A_i| > 2 \text{ (second condition)} \end{cases} \quad (4.2)$$

then A is a $\frac{4}{7}$ -GMMS allocation for the items in $\bigcup_{i \in N} A_i$ and the set of agents N .

Proof. Fix an agent a . Scale the valuation of the items so that $v_a(A_a) = 1$. Let $S := \{j \in A_i : i \in N, |A_i| = 2\}$. We have that $v_{aj} \leq v_a(A_a) = 1$ for $j \in S$ (condition 1). Consider the subset of agents $N' \subseteq N$ with $|N'| = n'$ and $a \in N'$. Let $M' = \bigcup_{i \in N'} A_i$. We will prove that $\mu_a^{n'}(M') < \frac{7}{4}$. We may assume that there is no agent $j \neq i$ in N' with $|A_j| = 1$ (Corollary 4.3). Assume that in N' , there are k agents with a size two bundle ($|S \cap M'| = 2k$) and hence there are $n' - k - 1$ agents with a bundle of size three or more.

- Case 1: $|S \cap M'| \leq n'$ then we have

$$v_a(M') \leq 2k + \frac{3}{2}(n' - k - 1) + 1 \leq \frac{1}{2}(k) + \frac{3}{2}(n') - \frac{1}{2} \leq \frac{7}{4}(n') - \frac{1}{2} \quad (4.3)$$

It implies that

$$\mu_a^{n'}(M') < \frac{7}{4}.$$

- Case 2: $|S \cap M'| = n' + t$ for $t = 1, \dots, n' - 2$ then we prove that $\mu_a^{n'}(M') < \frac{7}{4}$ by induction on t .

- If $t = 1$ ¹, in $P_a^{n'}(M')$ there will be at least one bundle with two items from S (pigeonhole principle). Let us say $j_1, j_2 \in M'$ are those items which are in the same bundle. Then we have

$$\mu_a^{n'}(M') \leq \mu_a^{n'-1}(M' \setminus \{j_1, j_2\}) \leq \frac{v_a(M' \setminus \{j_1, j_2\})}{n' - 1}. \quad (4.4)$$

This is true because in $P_a^{n'}(M')$ the value of all bundles is at least $\mu_a^{n'}(M')$ including the $n' - 1$ bundles which do not contain items j_1 and j_2 . Even though there might be more items in the bundle containing j_1 and j_2 , this case will only reduce the right hand side of the inequality (4.4). Therefore, the inequality still holds. If we look more closely to the first inequality in (4.4), the number of items in $S \cap (M' \setminus \{j_1, j_2\})$ is at most $n' - 1$ so we can use (4.3) and we get

$$v_a(M' \setminus \{j_1, j_2\}) \leq (n' - 1) + \frac{3}{2} \left(\frac{n' - 3}{2} \right) + 1 \leq \frac{7}{4}(n' - 1) - \frac{1}{2}$$

which is the previous case for $n' - 1$ ($n' - 1$ items from S , $\frac{n'-3}{2}$ of bundles with value of at least $3/2$, and items in A_a with total value of 1). Therefore,

$$\mu_a^{n'}(M') \leq \frac{7}{4}.$$

- for any $t > 1$ we can again claim that in MMS partition of agent a there will be a bundle of size two which we can set aside where the MMS value of current items and bundles is at most the MMS value of the remaining items which will give us the case for $t - 1$.

□

We use the conditions in (4.2) to get a $\frac{4}{7}$ -GMMS allocation.

4.5 Basics of the Algorithm

Before going to the details of the algorithm in the next section, let us introduce the bundles of the agents and the graph $G = (N, E)$ which is constantly used in Algorithm 11.

4.5.1 Bundle of the Agents

The bundle of each agent $i \in N$ consists of two sets A_i and R_i . A_i is the set which will be eventually assigned to agent i at the end of the algorithm. R_i is the set for holding the reserved item, i.e., R_i will have at most one item in all stages of the algorithm and the algorithm tries to empty this set using cycle removal function (see Algorithm 12).

4.5.2 The Graph

Consider the directed graph $G = (N, E)$ where N is the set of agents and $E = E_1 \cup E_2$. Basically, E_1 is the set of arcs which shows envy, i.e., $(a, b) \in E_1$ if $v_a(A_a) < v_a(A_b)$ so $G_1 = (N, E_1)$ is the well-known envy

¹It means that in $\{A_i : i \in N'\}$, besides A_a , there are $k = \frac{n'+1}{2}$ bundles of size two and $n' - k - 1 = \frac{n'-3}{2}$ bundles of size three or more.

graph. E_2 are the set of arcs added to G in order to check the second condition in (4.2). Before going to description of E_2 , for any i with $|R_i| = 1$ let

$$J_i := \begin{cases} A_i & \text{if } |A_i| \leq 2 \\ \text{a minimal } J' \subseteq A_i : v_k(J' \cup R_i) > 3/2v_k(A_k) \text{ for some } k \in N & \text{if } |A_i| > 2 \end{cases} \quad (4.5)$$

which means that J_i is a minimal subset of A_i where $J_i \cup R_i$ violates the second condition of (4.2) for some agent $k \in N$. J_i can be obtained greedily by setting $J_i \leftarrow A_i$ and removing items from J_i one by one until if we go further the value of $J_i \cup R_i$ reduces to at most $3/2v_k(A_k)$ for all $k \in N$. We define E_2 as

$$(a, b) \in E_2 \text{ if } v_a(J_b \cup R_b) > 3/2v_a(A_a). \quad (4.6)$$

Using the cycle removal function, we keep the graph G constantly acyclic.

The graph G is made by adding extra arcs (E_2) to the envy graph. Therefore, if an agent is a source agent in G it is also a source agent in the envy graph.

4.6 Algorithm Description

In this section, we explain Algorithm 11. The algorithm starts with empty bundles, i.e., $A_i = R_i = \emptyset \forall i \in N$ and an arbitrary order of the agents. Starting from the first agent, say agent 1, in the order, the algorithm adds her favorite item in M to A_1 and removes the assigned item from M . Then it repeats for the next agent in the order until all the agents have one item which they prefer to any remaining item in M (throughout the chapter, M is used as the set of unassigned items). Next, it starts iterations and each iteration has three main parts:

1. Choosing a source agent (which is always possible since G is acyclic).
2. Adding/reserving an item to the source agent.
3. Removing cycles.

Throughout the algorithm

1. $A = (A_1, \dots, A_n)$ holds the conditions in (4.2) and
2. at each round of the algorithm, $v_i(A_i)$ will never decrease for any $i \in N$.

Let P be the set of agents who can violate the second condition in (4.2) if we add an item to some source agent k , i.e.,

$$P := \{i \in N : \exists j \in M \text{ which } v_i(A_k \cup \{j\}) > 3/2v_i(A_i) \text{ for some source agent } k \text{ with } |A_k| \geq 2\}.$$

Also, let B be the set of favorite un-allocated items of agents in P , i.e.,

$$B := \{j \in M : j \text{ is the favorite item of agent } i \text{ in } M \text{ for } i \in P\}.$$

4.6.1 Choosing the source agent

We do not add the new item to *any* source agent. Instead, we choose an agent who helps us the most to keep the conditions in (4.2). Let $\mathcal{S} := \{i \in N : i \text{ is a source in } G\}$ be the set of agents who are a source in G . We pick the agents with the following priorities.

1. *Type 1:* $i \in \mathcal{S} : |A_i| = 1$. After adding another item to A_i , the size of A_i will become two. For any agent k , $v_k(A_k)$ is greater than both of the items in A_i since agent i is the source agent and we will prove in Lemma 4.10 that any un-allocated item has value less than $v_k(A_k)$ for all the agents. Therefore, conditions in (4.2) will not be violated.
2. *Type 2:* $i \in \mathcal{S} : \exists j \in B : v_k(A_i \cup \{j\}) \leq 3/2v_k(A_k) \forall k \in P \setminus \{i\}$. If such item j and agent i exists we add the item j to A_i and the conditions in (4.2) will still hold for A_i .
3. *Type 3:* $i \in \mathcal{S} : \text{there exists an agent in } P \text{ which is reachable from } i \text{ using a directed path in } G$. In this case we add the item to R_i . We will prove that if there are no agent type (1) and (2), there will exist an agent of type (3).

Lemma 4.6. *If there is no source agent of type 1 and 2, there will be a type 3 source agent.*

Proof. Since there is no type 2 source agent, we can conclude that $P \neq \emptyset$. Pick one agent k from P . Either k is a source agent (then it is reachable from k) or there is path from a source agent to k . We can find such source agent by going to the opposite direction of an incoming arc to k and continue going backward until we reach a node which has no incoming arcs (a source agent). That source agent is a type 3 source agent because there is no type 1 and 2 source agents and there is a path from the source agent to k . $\square$

4.6.2 Adding/reserving an item to the source agent

After choosing agent i who is candidate to get an item, the algorithm adds an item from S according to the priority type of the agent i :

1. If i is a type 1 source, j is favorite item of agent i in B and it will be added to A_i .
2. If i is a type 2 source, pick $j \in B$ such that $v_k(A_i \cup \{j\}) \leq 3/2v_k(A_k)$ for all $k \in P$ (or simply, by definition of P we can say for all $k \in N$), if there are multiple such items the one which is i 's favorite will be chosen and it will be added to A_i .
3. If i is a type 3 source, j is favorite item of agent i in B and it will be added to R_i .

4.6.3 Updating the graph and the reserved bundles

Updating the graph G .

In each round, after adding an item, the graph needs to be updated.

If the item is added to A_i for some i , the value of A_i increases for all agents. Therefore, it might create some incoming E_1 edges to node i (note that if an item is being added to A_i it implies that i is a source agent and $R_i = \emptyset$. This is because all agents with $R_i \neq \emptyset$ have incoming E_2 edges.) and, also, some of outgoing E_1 and/or E_2 edges from i might need to be removed.

If the item is added to R_i for some i , it means that one or more agents value $A_i \cup R_i$ more than $3/2$ of their assigned bundle in $A = (A_1, \dots, A_n)$. Therefore, some incoming E_2 edges will be added to node i , but none of the outgoing edges from node i will be removed, because the value of A_i is not changed.

One more subtle change happens when an item is added to A_i with outgoing E_2 edge from i to k and agent i do not value the former $J_k \cup R_k$ more than $3/2$ of A_i . By definition of J_k in (4.5), J_k changes to a bigger subset of A_k where someone (agent i or another agent) value $J_k \cup R_k$ more than $3/2$ of their bundle in $A = (A_1, \dots, A_n)$. Consequently, new incoming E_2 edges will be added to node k . Moreover, if the value of A_i increase in a way that she does not value $A_k \cup R_k$ more than $3/2$ of A_i and no other agent value $A_k \cup R_k$ more than $3/2$ of their bundle in $A = (A_1, \dots, A_n)$, we should update R_k which will be explained in the next part.

Updating the reserved bundles R_i 's.

We mentioned above that if an item is added to A_i some outgoing edges from node i might need to be removed. Consider some node k with $R_k \neq \emptyset$ and hence incoming E_2 edges. If after some rounds in Algorithm 11 all the incoming edges to the node j is removed, it means that no one values $A_k \cup R_k$ more than $3/2$ of their assigned bundle in $A = (A_1, \dots, A_n)$. Therefore, we can merge R_i to A_i , i.e., $A_i \leftarrow A_i \cup R_i$ and $R_i \leftarrow \emptyset$. This action will increase the value of A_k and hence we update G as explained in the previous part.

4.6.4 Removing cycles

The graph which the algorithm works with is slightly different from the normal envy graph. Consider $G = (N, E)$ which is introduced in Section 4.5.2. At each round, after adding an item to one of the bundles if a cycle appears we remove them using the Algorithm 12. Before going to the details we will study the graph G more carefully.

For all agents $i \in N$ with $|R_i| = 1$, the last item which is added to their bundle, $A_i \cup R_i$, is the item in R_i . Therefore, no agent values A_i more than their own bundle since i was a source agent before adding the item to R_i . Moreover, the incoming arc(s) to i are from E_2 .

After adding an item to R_i for some source agent $i \in N$, at least one incoming arc, (k, i) , will be added to agent i from some agent k . If there is a directed path from i to k in G , the added arc will make a cycle. Otherwise, it will reduce the number of source agents in G by one.

Cycle removal happens in two steps

1. *Prune*: Pruning bundles if necessary
2. *Exchange*: Exchanging bundles

We remove one cycle at a time. Assume C is a cycle in G . Without loss of generality, let $1 \rightarrow 2 \rightarrow 3 \rightarrow \dots \rightarrow c \rightarrow 1$ be the order of vertices in C and for simplicity of the notation let $0 \equiv c$. Each arc $(i-1, i)$ in C either belongs to E_1 or E_2 . If $(i-1, i) \in E_2$ we need to prune the bundle $(A_i \cup R_i)$ before exchanging.

Prune. For all $(i-1, i) \in E_2$ let j_{i-1}^i be the least valued item of agent $i-1$ in the set J_i (if there are many such items we pick one arbitrarily). Prune A_i to be $(J_i \cup R_i) \setminus \{j_{i-1}^i\}$ and set $R_i = \emptyset$ (we add the remaining items, $A_i \setminus J_i$ and j_{i-1}^i , back to M).

Lemma 4.7. *The value of a pruned bundles with at least three items is at most $3/2v_i(A_i)$ for all $i \in N$.*

Proof. Directly from the definition of J_i in (4.5). □

Exchange. In the second step of cycle removal, each agent gives her bundle to the precedent vertex and gets the bundle of the next vertex.

Lemma 4.8. *After each cycle removal, value of no agent for their allocated bundle will decrease.*

Proof. In removal of each cycle C with order of the nodes $1 \rightarrow 2 \rightarrow 3 \rightarrow \dots \rightarrow c \rightarrow 1$, there are three different type of agents.

1. Agents who are not C : their bundle will remain unchanged.
2. Agents in C who are starting point of arc $(i, i+1) \in E_1$ for $i+1 \in C$: they will receive $(A_{i+1} \cup R_{i+1})$ which they were envying so their valuation will increase.
3. Agents in C who are starting point of arc $(i, i+1) \in E_2$ for $i+1 \in C$: they will receive $A_i \leftarrow (J_{i+1} \cup R_{i+1}) \setminus \{j_i^{i+1}\}$. Note that the last item added to the bundle of agent $i+1$ is the item in R_{i+1} and before adding that item agent $i+1$ is a source agent. Therefore, we have

$$v_i(A_{i+1}) \leq v_i(A_i) \quad (4.7)$$

which implies that $v_i(J_{i+1}) \leq v_i(A_i)$ because by definition $J_{i+1} \subseteq A_{i+1}$. From (4.7) we can conclude that $v_i(\{j_i^{i+1}\}) \leq \frac{1}{2}v_i(A_i)$. Moreover, according to (4.6) we have that

$$v_i(J_{i+1} \cup R_{i+1}) > 3/2v_i(A_i).$$

Therefore, $v_i((J_{i+1} \cup R_{i+1}) \setminus \{j_i^{i+1}\}) > v_i(A_i)$.

□

Theorem 4.9. *Let $A = (A_1, \dots, A_n)$ be the output of Algorithm 11. Then for $a, i \in N$ with $i \neq a$ we have*

$$v_a(A_i) \geq \begin{cases} \max_{j \in A_i} v_{aj} & |A_i| = 2 \text{ (first condition)} \\ \frac{2}{3}v_a(A_i) & |A_i| > 2 \text{ (second condition)}. \end{cases} \quad (4.2)$$

We will prove Theorem 4.9 by proving for each condition in (4.2) separately. Before going to the proof of the theorem we need to first prove the following lemma.

Lemma 4.10. *At any stage of the Algorithm 11, for any agent $i \in N$ and any un-allocated item $j \in M$, we have that*

$$v_i(A_i) \geq v_{ij}.$$

Proof. Any un-allocated item $j \in M$ either it has never been allocated to an agent or it has been moved back to M as a result of a pruning in cycle removal function.

If j has never been allocated, v_{ij} is less than the value of the first item added A_i . In the next stages of the algorithm the value of A_i will never decrease. Because either a new item is added to A_i or it goes to a cycle removal process and in both cases $v_i(A_i)$ will not decrease (see Lemma 4.8 for the value of $v_i(A_i)$ after cycle removal).

If j is removed from a bundle after cycle removal, it means that j was belonged to $A_{i'}$ for some source agent i' (see Section 4.6.4 for more details). It implies that $v_i(A_i) \geq v_{ij}$ because agent i did not envy agent i' when j was in $A_{i'}$. □

Proof of Theorem 4.9. • *Condition 1.* Fix agent a and i with $|A_i| = 2$. There are two different scenarios for agent i .

- In the first scenario, agent i was a source agent with only one item j_1 and a second item, say j_2 , is added to her bundle. In this case, $v_a(A_a) \geq v_{j_1}$ since agent i was a source agent and $v_a(A_a) \geq v_{j_2}$ by Lemma 4.10.
- In the second scenario, A_i is a result of a pruning so one item used to belong to A_i and the other item used to belong to R_i . As it is mentioned earlier in Section 4.6.4, if $|R_i| = 1$, it implies that $v_a(A_a) \geq v_a(A_i)$ and the item in R_i is the last item added to the bundle. Therefore, $v_a(A_a)$ is greater than both items in A_i .
- *Condition 2.* A new item is added to A_i in two situations. Either agent i is a type 1 or 2 source agent which means that the condition 1 and 2 will be held. Or, agent i gets a pruned bundle which according to Lemma 4.7 the pruned bundle satisfies the condition 2.

□

Corollary 4.11. *After Algorithm 11 stops, all A_i 's hold the conditions in (4.2) but there might still be some R_i 's for $i \in N$ which are not empty. We can donate these items and get a $\frac{4}{7}$ -GMMS Algorithm.*

4.7 Conclusions

In this chapter, we studied the GMMS allocations when agents have additive valuations. We showed an algorithm which gives a $\frac{4}{7}$ -GMMS allocation after removing at most n items. Chaudhury et. al [24] independently studied this problem and introduced an algorithm where it gives a $\frac{4}{7}$ -GMMS allocation with no need to discard any items. The existence of a better bound for GMMS allocations is still an open problem.

Algorithm 11: $\frac{4}{7}$ -GMMS Algorithm

Input: $I = \langle N, M, V \rangle$ **Output:** $A = (A_1, \dots, A_n)$ which is $\frac{4}{7}$ -GMMS

```
1 for  $i \in N$  do
2    $j \leftarrow i$ 's favorite item in  $M$ ,  $R_i \leftarrow \emptyset$  // initializing  $R_i$  and  $A_i$ 
3    $A_i \leftarrow \{j\}$ 
4   remove  $j$  from  $M$ 
5  $P := \{i \in N : \exists j \in M \text{ which } v_i(A_k \cup \{j\}) > 3/2v_i(A_i) \text{ for some source agent } k \text{ with } |A_k| \geq 2\}$ 
6  $B := \{j \in M : j \text{ is favorite item of agent } i \text{ in } M \text{ for } i \in P\}$ 
7  $G := (N, E)$  where  $E = E_1 \cup E_2$ 
8  $(a, b) \in E_1$  if  $v_a(A_a) < v_a(A_b)$ 
9 for all source agent  $i$  in  $G = (N, E)$ :
    $J_i := \begin{cases} A_i & \text{if } |A_i| \leq 2 \\ \text{minimal } J' \subseteq A_i : v_k(A_k) < 3/2v_k(J' \cup R_i) \text{ for some } i \in N & \text{if } |A_i| > 2 \end{cases}$ 
10  $(a, b) \in E_2$  if  $v_a(J_b \cup R_b) > 3/2v_a(A_a)$ 
11 while  $|M| > 0$  do
12   if  $|P| > 0$  then
13     pick a source agent  $i$  in  $G$  with these priorities: (1) type 1:  $|A_i| = 1$  (2) type 2: there exist
        $j \in B : v_k(A_i \cup \{j\}) \leq 3/2v_k(A_k)$  for all  $k \in P \setminus \{i\}$  (3) type 3: there exist an agent in  $P$ 
       which is reachable from  $i$  using a directed path in  $G$ 
14     if  $i$  is type 1 then
15        $j \leftarrow i$ 's favorite item in  $B$ 
16       add  $j$  to  $A_i$ 
17     else if  $i$  is type 2 then
18        $j$  is an item in  $B$  which  $v_k(A_i \cup \{j\}) \leq 3/2v_k(A_k)$  for all  $k \in P \setminus \{i\}$  // if there are
       multiple such items choose the one which is  $i$ 's favorite
19       add  $j$  to  $A_i$ 
20     else
21        $j$  is an item from  $S$ 
22       add  $j$  to  $R_i$ 
23     remove  $j$  from  $M$ 
24     update  $G$  and  $R_i$ 's
25     REMOVE CYCLES
26   else
27     pick a source agent  $i$ 
28      $j \leftarrow i$ 's favorite item in  $M$ 
29     remove  $j$  from  $M$ 
30     update  $G$  and  $R_i$ 's
31     REMOVE CYCLES
```

Algorithm 12: REMOVE CYCLES

Input: $I = \langle N, M, V \rangle$

Output: $A = (A_1, \dots, A_n)$ which is $\frac{4}{7}$ -GMMS

```
1 while there exist a cycle in G do
2   pick a cycle  $C$  arbitrarily
3   for  $i \in C$  and  $|R_i| = 1$  do
4     let  $i'$  be the precedent node of  $i$  in  $C$ 
5      $j \leftarrow i'$ 's least favorite item in  $J_i$ 
6      $A_i \leftarrow (J_i \setminus \{j\}) \cup R_i$  // Prune
7     add removed items to  $M$ 
8      $R_i \leftarrow \emptyset$ 
9   every agent in  $C$  gets the bundle of their successor agent // Exchange
```

Chapter 5

MMS Value in Mixed Manna with Restricted Valuations and Constant n

5.1 Introduction

As it was mentioned in earlier chapters, finding MMS value of the agents is NP-hard, even for the good manna and chore manna. However, for either of these settings, there is a PTAS to obtain MMS value [48] (good manna) and [34] (chore manna). Such a PTAS has served as an important tool to several works to study the MMS problem. In good manna [40, 39, 6, 31, 29] use a PTAS for finding MMS values in order to find an α -MMS allocation for various constant α 's. Similarly, in chore manna [33] uses a PTAS for MMS values as a subroutine to find PTAS for 11/9-MMS allocation.

In the following two chapters, we consider a *mixed manna*, i.e., a set M of discrete items that are goods/chores, among a set N of agents with additive valuations. A mixed manna allows an item to be a good (positively valued) for some agents, and a chore (negatively valued) for others. We specifically study the problem of finding the MMS value of the agents. Note that finding the MMS value of an agent is equivalent to finding an MMS allocation when all agents have valuations identical to this agent (See Definition 2.1).

Surprisingly, [38] showed that even in a highly restricted setting of two identical agents where the mixed manna has only two chores, it is NP-hard to find *approximate* MMS values within *any* constant factor (Theorem 5.3).

To tackle this issue the MMS problem for mixed manna has to be studied in restricted settings. Specifically, in this chapter we design a PTAS that returns $(1 - \epsilon)$ approximate MMS values of agents when the following two conditions hold (i) number of agents n is a constant, and (ii) for every agent i , her total (absolute) value for all the items ($|v_i(M)|$) is significantly greater than the minimum of her total value of goods (v_i^+) and her total (absolute) value for chores (v_i^-), i.e., for a constant $\tau > 0$, $|v_i(M)| \geq \tau \cdot \min\{v_i^+, v_i^-\}$.

Kulkarni et al. [38] show that if either condition is not satisfied, then finding MMS value approximately is NP-hard within *any* non-trivial factor even when there are only *two agents* and M has only two chores.

5.2 Problem Definition and Notations

In Chapter 2, given an MMS instance $\mathcal{I} = \langle N, M, V \rangle$ we defined the MMS value and MMS partition of an agents (Definition 2.1), as well as the MMS allocation, α -MMS allocation, and the MMS problem (Definition

2.2). We use $[k]$ to denote the set $\{1, 2, \dots, k\}$.

Given an MMS instance $\mathcal{I} = \langle N, M, V \rangle$, in order to design a PTAS to approximately find MMS value of an agent $i \in N$, we find a partition where all parts value at least $(1 - \epsilon)\mu_i$ for agent i .

In order to capture identity of the agents, we tweak the notations from Chapter 2. Instead of μ_i , the MMS value of the identical agents is shown by μ . We also define the MMS instance as below.

Definition 5.1 (MMS instance with identical agents). *An MMS instance with identical agents is a tuple $\mathcal{I} = \langle n, M, v \rangle$, where n is the number of identical agents, M is a set of m indivisible items, and $v : 2^M \rightarrow \mathbb{R}$ is the identical additive valuation function of all agents, represented by $v(S) = \sum_{j \in S} v_j$ for $S \subseteq M$.*

It is easy to check that when agents are identical, MMS allocation, i.e., α -MMS allocation for $\alpha = 1$, always exists because each MMS partition is also an MMS allocation.

Proposition 4. *Finding an MMS partition of an agent $i \in N$, is equivalent to the finding α -MMS allocation for $\alpha = 1$ when all the agents have valuations identical to the agent i .*

In other words, given $\mathcal{I} = \langle n, M, v \rangle$, we are interested to find $(1 - \epsilon)$ -MMS allocation when the agents are identical. For this reason, in the following two chapters, the notions of MMS partition and MMS allocation are used interchangeably.

Items of a mixed manna can be divided into two sets. *Goods* are the items valued positively according to v . The set of goods is denoted by $M^+ = \{j \in M \mid v_j \geq 0\}$. *Chores* are items valued negatively, and the set of chores is termed $M^- = \{j \in M \mid v_j \leq 0\}$. We denote by v^+ the sum of values of all goods in the manna. That is, $v^+ = \sum_{j \in M^+} v_j$. Similarly, we denote by v^- the sum of absolute values of all chores, i.e., $v^- = \sum_{j \in M^-} |v_j|$.

The set of items valued non-negatively (negatively) are called her *Goods* (*Chores*), and denoted by $M^+ = \{j \mid v_j \geq 0\}$ ($M^- = \{j \mid v_j < 0\}$).

Because we are studying the mixed manna setting, MMS value of the agents (μ) can be negative. Therefore, we need to modify the definition of α -MMS allocation (2.2) to apply for negative MMS values as well.

Definition 5.2 (α -MMS allocation). *A is called an α -MMS allocation, if for a given $\alpha > 0$, for each agent $i \in N$ we have $v_i(A_i) \geq \alpha\mu_i$ if $\mu_i \geq 0$, and $v_i(A_i) \geq (1/\alpha)\mu_i$, if $\mu_i < 0$. Equivalently,*

$$v_i(A_i) \geq \min\{\alpha\mu_i, (1/\alpha)\mu_i\}.$$

Kulkarni et al. [38] showed that in mixed manna finding an α -MMS allocation is NP-hard for any constant α .

Theorem 5.3 (Theorem 3.1 Kulkarne, Mehta, and Taki [38]). *For any instance $\mathcal{I} = \langle n, M, v \rangle$ with identical agents and $v(M) > 0$ such that exactly one of the following two holds: (a) either $n = 2$ or (b) $|v(M)| \geq \tau \cdot \min\{v^+, v^-\}$ for a constant τ , finding an α -MMS allocation of $\mathcal{I} = \langle n, M, v \rangle$ for any $\alpha \in (0, 1]$ is NP-hard.*

Since the problem of finding α -MMS allocation is NP-hard for any $\alpha \in (0, 1]$, we design a PTAS to compute a $(1 - \epsilon)$ -MMS allocation for a sub-class of instances. To characterize this sub-class, we will make the following two assumptions for $\mathcal{I} = \langle n, M, v \rangle$.

1. the number of agents n is constant, and
2. for some constant $\tau > 0$, for every agent $i \in N$, $|v(M)| \geq \tau \cdot \min\{v^+, v^-\}$,

In the following two sections we prove the following theorem.

Theorem 5.4. *Given an instance $\mathcal{I} = \langle n, M, v \rangle$ and a constant $\epsilon > 0$, if (i) n is a constant and (ii) for each $i \in [n]$, $|v(M)| \geq \tau \cdot \min\{v_i^+, v_i^-\}$, where $\tau > 0$ is a constant. Then, there is a PTAS to compute a $(1 - \epsilon)$ -MMS allocation.*

5.3 Finding MMS values of Agents When $\mu \geq 0$

In this section we prove Theorem 5.4 for the case when $\mu \geq 0$, i.e., given an instance $\mathcal{I} = \langle n, M, v \rangle$, we describe an algorithm to find a $(1 - \epsilon)$ -MMS allocation for any constant $\epsilon > 0$. Using scale invariance (Lemma 2.5), here on we assume $v(M) = n$ without loss of generality. Due to Lemma 2.4, this implies $\mu \leq v(M)/n = 1$.

The high level ideas used in the algorithm are as follows. First is a classification of all the items into two sets, **Big** and **Small**, based their value (Section 5.3.1). Using this we prove that $|\text{Big}|$ is constant, which allows the enumeration of all allocations of **Big**, referred as *partitions* of **Big** to avoid confusion with allocations of M . Next in Section 5.3.2 we explain a short procedure which allows us to characterize partitions of **Big** as *valid* or *invalid*. We show that there is at least one valid partition corresponding to which there is an MMS allocation, hence all invalid partitions can be discarded. Finally in Section 5.3.3, we describe a sub-routine called Bag-Fill, that greedily allocates or ‘fills’ the items from **Small** upon partitions or ‘bags’ of **Big** that satisfy certain constraints to obtain a $(1 - \epsilon)$ -MMS allocation. The main algorithm (Section 5.3.4) enumerates all partitions of **Big**, discards the invalid partitions, and applies Bag-Fill if its constraints are satisfied. If the constraints are not satisfied, we show that we can apply the PTAS for obtaining MMS allocations with identical agents for a goods manna, and obtain a $(1 - \epsilon)$ -MMS allocation.

We now discuss these key ideas formally followed by the algorithm in separate subsections.

Lemma 5.5 helps to find the sign of MMS value of the agents and it will be useful in what follows.

Lemma 5.5. $v(M) \geq 0$ iff $\mu \geq 0$.

Proof. If the sum of valuations of all items $v(M)$ is negative, there can be no partition where every part has non-negative valuation. Hence, μ is negative. If the sum of valuations is positive, then adding all items to one bundle and no item in other bundles makes the least-valued bundle have zero value. Thus, in this case, $\mu \geq 0$. $\square$

5.3.1 Big and Small items

Given an instance $\mathcal{I} = \langle n, M, v \rangle$ and a constant $\epsilon > 0$, let **Big** be the set of items in M which have absolute value higher than $\frac{\epsilon}{2}$, i.e.,

$$\text{Big} = \{j \in M : |v_j| \geq \frac{\epsilon}{2}\}.$$

Let Big^+ and Big^- respectively be the sets of the goods and the chores in **Big**, i.e., $\text{Big}^+ = \text{Big} \cap M^+$, and $\text{Big}^- = \text{Big} \cap M^-$. Let **Small** be the set of small items, i.e., $M \setminus \text{Big}$, and similarly define the sets of goods (Small^+) and chores (Small^-) in **Small**.

We abuse notation slightly and call items in the set **Big** (or **Small**) as **Big** (resp. **Small**) items.

Lemma 5.6. $|\text{Big}| = O(n/\epsilon)$.

Proof. As $v(M) = n$, we have $n = v(M^+) - |v(M^-)|$. Then, as $v(M^+) \geq (1 + \tau)|v(M^-)|$,

$$n \geq v(M^+) - \frac{v(M^+)}{1 + \tau} \implies v(M^+) \leq \frac{n(1 + \tau)}{\tau}. \quad (5.1)$$

Finally, by the definition of Big^+ we have

$$|\text{Big}^+| \leq \frac{v(M^+)}{\frac{\epsilon}{2}} \leq \frac{2n(1+\tau)}{\epsilon\tau}.$$

Similarly, we have

$$n \geq (1+\tau)|v(M^-)| - |v(M^-)| \Rightarrow |v(M^-)| \leq n/\tau. \quad (5.2)$$

Thus, the number of Big chores is bounded as $|\text{Big}^-| \leq \frac{2n}{\epsilon\tau}$. Hence, $|\text{Big}| = |\text{Big}^+ \cup \text{Big}^-| = O(\epsilon)$. $\square$

As n and ϵ are constant, Lemma 5.6 implies that all partitions of Big can be enumerated in constant time.

5.3.2 Valid and Invalid partitions of Big

Given a partition $A = \{A_1 \cdots, A_n\}$ of M , denote by $B = \{B_1 \cdots, B_n\}$ the partition from $M \setminus \text{Small}^+$ where each $B_k = A_k \cap \text{Big}$. Classify the bundles of A based on their value from B into sets $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$, and $\mathcal{B}_4$, together called the $\mathcal{B}$ -sets, as follows.

$$\begin{aligned} \mathcal{B}_1 &:= \{A_k \in A : v(B_k) > 1\} & \mathcal{B}_2 &:= \{A_k \in A : 1 - \epsilon \leq v(B_k) \leq 1\} \\ \mathcal{B}_3 &:= \{A_k \in A : 0 \leq v(B_k) < 1 - \epsilon\} & \mathcal{B}_4 &:= \{A_k \in A : v(B_k) < 0\} \end{aligned} \quad (5.3)$$

We will abuse notation to denote all items in the sets in any $\mathcal{B}_i$, i.e. $\cup_{A_k \in \mathcal{B}_i} A_k$, by $\mathcal{B}_i$.

Given a partition of Big B , we now explain a procedure using which we classify the partition as valid or invalid. First classify the bundles of B into four sets as per equation (5.3). Initially, all Small items are unallocated. Then while $\mathcal{B}_1 \neq \emptyset$ and $\text{Small}^- \neq \emptyset$, assign any item from Small^- to any agent that has a bundle from $\mathcal{B}_1$. Re-classify the bundles using equation (5.3) and remove the assigned item from Small^- after every assignment. This procedure ends when either $\mathcal{B}_1$ or Small^- becomes empty, or both. If in the end $\mathcal{B}_1 \neq \emptyset$, and we also have (a) $\mathcal{B}_4 \neq \emptyset$ and (b) $v(\mathcal{B}_3 \cup \mathcal{B}_4 \cup \text{Small}^+) < (1 - \frac{\epsilon}{2})(|\mathcal{B}_3| + |\mathcal{B}_4|)$ then we call B *invalid*. All partitions of Big that are not invalid are called valid.

Lemma 5.7. *There exists an MMS partition where the Big items are partitioned according to a valid partition.*

Proof. Let A be an MMS partition with the lowest value of $|\mathcal{B}_1| + |\mathcal{B}_4|$, and let B be its corresponding partition of Big . Suppose B is invalid. Then the agents with bundles from $\mathcal{B}_3$ or $\mathcal{B}_4$ can only receive items from $\mathcal{B}_3 \cup \mathcal{B}_4 \cup \text{Small}^+$ in A . From Lemma 2.4 and the definition of invalid partitions giving $v(\mathcal{B}_3 \cup \mathcal{B}_4 \cup \text{Small}^+) < (1 - \frac{\epsilon}{2})(|\mathcal{B}_3| + |\mathcal{B}_4|)$, we have $\mu < (1 - \frac{\epsilon}{2})$.

As both $\mathcal{B}_1, \mathcal{B}_4 \neq \emptyset$, consider any bundles B and B' respectively from $\mathcal{B}_1$ and $\mathcal{B}_4$. Let $v(B) = (1+x)$ and $v(B') = -y$ for some $x, y > 0$. In A , B' must be bundled with a set of items from Small^+ , denoted as $\mathcal{S}^+$, of value at least $y + \mu$. We form two new bundles A_1 and A_2 of at least MMS value using B, B' and $\mathcal{S}^+$ as follows. First merge B and B' into one bundle (with value $1+x-y$). If $1+x-y \geq \mu$, call this bundle A_1 and add all the remaining items from $\mathcal{S}^+$ to A_2 . Each bundle thus has value at least μ . Otherwise, if $1+x-y < \mu$, add items from $\mathcal{S}^+$ one by one, each time to the bundle with the lower value before adding the item. Let A_1 and A_2 be the resulting bundles after adding all items in $\mathcal{S}^+$. Without loss of generality, let $v(A_1) \geq v(A_2)$. As each item in $\mathcal{S}^+$ has value at most $\frac{\epsilon}{2}$ and is always added to the lower valued bundle,

$v(A_1) - v(A_2) \leq \frac{\epsilon}{2}$. Thus,

$$\begin{aligned} v(A_1) + v(A_2) &\geq 1 + x - y + y + \mu = 1 + x + \mu \text{ and } v(A_1) - v(A_2) \leq \frac{\epsilon}{2} \\ \implies v(A_2) &\geq \frac{1}{2}(1 + x + \mu - \frac{\epsilon}{2}) > \frac{1}{2}(1 + \mu - \frac{\epsilon}{2}) > \mu \implies v(A_1) > \mu. \end{aligned}$$

No item from **Big** is assigned to A_2 . Thus, $A_2 \notin \mathcal{B}_1 \cup \mathcal{B}_4$, and A_1 and A_2 combined with the allocations of the remaining agents who did not get B or B' in A form an MMS allocation with a smaller value of $|\mathcal{B}_1| + |\mathcal{B}_4|$ than A , a contradiction. Thus, B is valid. $\square$

5.3.3 Algorithm Bag-Fill

In this section we design the algorithm Bag-Fill (Algorithm 13) that generalizes Bag-filling algorithms in Chapter 2, as well as the Bag-filling algorithms in [31, 28, 29] to the mixed setting. Bag-Fill (Algorithm 13) takes as input an MMS instance $\mathcal{I} = \langle n, M, v \rangle$, and a partition of the **Big** items of M , denoted by $B = \{B_1, B_2, \dots, B_n\}$ such that they satisfy one of the two condition sets (5.4) or (5.5). It outputs an allocation of items $A = \{A_1, \dots, A_n\}$ where $v(A_i) \geq 1 - \epsilon$, for all $i \in [n]$.

$$\begin{aligned} v(\text{Small}) + \sum_{k=1}^n v(B_k) &\geq n. & v(\text{Small}) + \sum_{k=1}^n v(B_k) &\geq n(1 - \frac{\epsilon}{2}). \\ v(B_k) &\leq 1 \quad \forall k \in [n]. & v(B_k) &\leq 1 - \frac{\epsilon}{2} \quad \forall k \in [n]. \\ |v_j| &< \epsilon \quad \forall j \in \text{Small}. & |v_j| &< \frac{\epsilon}{2} \quad \forall j \in \text{Small}. \end{aligned} \tag{5.4} \tag{5.5}$$

Algorithm 13 works as follows. It has $n - 1$ rounds. Each round starts with a bundle ('bag') from B . If the bag is valued at least $(1 - \epsilon)$, then it is assigned to some agent. If not, we first add all the unallocated **Small** chores to this bag. Then one by one we add the unallocated goods from **Small** until it is valued at least $(1 - \epsilon)$, and assign to some agent. After all rounds are done, in the last step, all remaining items from **Small** are added to the bag B_n . The next lemma proves the correctness of the algorithm.

Algorithm 13: Bag-filling to find $(1 - \epsilon)$ -MMS allocation of identical agents

Input : $\mathcal{I} = \langle n, M, v \rangle$, Partition of **Big** $\subseteq M$: $B = \{B_1, B_2, \dots, B_n\}$. Input satisfies Condition set (5.4) or (5.5)

Output: $A = \{A_1, \dots, A_n\}$ such that $v(A_i) \geq 1 - \epsilon$, $\forall i \in [n]$.

```

1  $A \leftarrow \emptyset$ ,  $\text{Small} \leftarrow M \setminus \text{Big}$ 
2 for  $k \in \{1, \dots, n - 1\}$  do
3   while  $v(B_k) < 1 - \epsilon$  do
4      $j \leftarrow \operatorname{argmin}_{j \in \text{Small}} v_j$ 
5      $B_k \leftarrow B_k \cup \{j\}$ ,  $\text{Small} \leftarrow \text{Small} \setminus \{j\}$ 
6    $A \leftarrow A \cup \{B_k\}$ 
7  $B_n \leftarrow B_n \cup \text{Small}$ ,  $A \leftarrow A \cup \{B_n\}$ 
8 return  $A$ 
```

Lemma 5.8. *If an MMS instance with identical agents satisfies condition set (5.4) or (5.5), then Algorithm 13 gives a $(1 - \epsilon)$ -Allocation.*

Proof. By induction on $k \in \{0, 1, 2, \dots, n-1\}$, we prove that the value of each assigned bundle after k rounds is in $[1 - \epsilon, 1]$ if the instance satisfies condition set (5.4), and in $[1 - \epsilon, 1 - \epsilon/2]$ if it satisfies condition set (5.5). The base case when $k = 0$ is trivial.

First consider the case when condition set (5.4) is satisfied. Assume the value of all bundles assigned to the first $k-1$ agents are in this range. Now $v(B_j) \leq 1$ for all $j \in \{k, \dots, n\}$. If $v(B_k) \geq 1 - \epsilon$, we are done. If not, then while the value of B_k is less than $1 - \epsilon$, the value of the unallocated items from **Small** is at least the value of all items minus that of all the allocated bundles and unallocated bags of **Big** items. This can be bounded as,

$$v(M) - \sum_{i < k} v(A_i) - v(B_k) - \sum_{i > k} v(B_i) > n - (k-1) - (1 - \epsilon) - (n-k) > \epsilon.$$

Hence, there is at least one unallocated **Small** good. Before adding the last **Small** good to A_k , its value was strictly less than $1 - \epsilon$. Adding the last item increases the value by at most ϵ . Hence, the value of A_k is at most 1. Thus, $v(A_k) \in [1 - \epsilon, 1]$, for all $k \in [n-1]$. As the total value of all items is at least n , and the total value of the $n-1$ assigned bundles is at most $n-1$, the last agent also gets a bundle of value at least 1.

Now suppose the instance satisfies condition set (5.5). In every round, while this is not true, the value of unallocated goods is at least $\epsilon/2$. Thus, there is at least one **Small** good. Finally, after assigning $n-1$ bundles, the total value remaining is at least $n(1 - \epsilon/2) - (n-1)(1 - \epsilon/2)$, hence the last bundle also has value at least $(1 - \epsilon)$. $\square$

5.3.4 The PTAS

We use the notions from the previous subsections and derive the PTAS shown in Algorithm 14. The PTAS works as follows. It first enumerates all the partitions of **Big**. For each partition B , it first classifies the bundles into the $\mathcal{B}$ -sets as per equation (5.3). If $\mathcal{B}_1$ is not empty, then add items from Small^- to any bag in $\mathcal{B}_1$, re-defining the sets and removing the assigned item from Small^- after each assignment. This process ends when either $\mathcal{B}_1 = \emptyset$ or $\text{Small}^- = \emptyset$. In the first case, condition set (5.4) of the Bag-Fill algorithm is satisfied, and we run Algorithm 13 (Line 8) and return its output.

Otherwise when $\mathcal{B}_1 \neq \emptyset$, if $|\mathcal{B}_4| = 0$, then reduce to the following goods manna α -MMS problem instance $\mathcal{I}' = \langle n', M', v' \rangle$. n' is the number of agents who received bundles from $\mathcal{B}_3$ or $\mathcal{B}_4$. M' has (a) Small^+ , with each item having the same value in v' as in v , and (b) for each bundle $B \in \mathcal{B}_3$, M' has a new item b with value $v'_b = v(B)$. Run the PTAS from [48] on $\mathcal{I}' = \langle n', M', v' \rangle$ to find a $(1 - \epsilon)$ -MMS allocation of M' among the $n' = n - (|\mathcal{B}_1| + |\mathcal{B}_2|)$ agents, and store its output in $\mathbb{A}$.

For the final case when $\mathcal{B}_1 \neq \emptyset$ and $|\mathcal{B}_4| > 0$, first check if the remaining unallocated goods and agents in $\mathcal{B}_3 \cup \mathcal{B}_4$ fulfill the condition set 5.5. If they do, apply the Bag-Fill and return the $(1 - \epsilon)$ -MMS allocation. If not, then B is invalid, hence discarded. After enumerating all the partitions, the algorithm returns the best allocation from $\mathbb{A}$.

Let us now discuss the analysis of the PTAS. To prove correctness when $\mathcal{B}_1 \neq \emptyset$ and $\mathcal{B}_4 = \emptyset$, we first show in Lemma 5.9 a relation between the MMS values of the given instance and the reduced goods manna instance. Let A^* be some MMS allocation, and $B^* = \{B_1^*, B_2^*, \dots, B_n^*\}$ be the allocation of **Big** items according to A^* .

Lemma 5.9. *If for B^* , the subsequent allocation of $\text{Big} \cup \text{Small}^-$ in Algorithm 14 has $\mathcal{B}_4 = \emptyset$, then,*

$$\mu^n(M) \leq \mu^{n-|\mathcal{B}_1|-|\mathcal{B}_2|} \left(\bigcup_{B \in \mathcal{B}_3} B \cup \text{Small}^+ \right).$$

Algorithm 14: $(1 - \epsilon)$ -MMS allocation for identical agents with $\mu \geq 0$

Input : $\mathcal{I} = \langle n, M, v \rangle$ such that $v(M) = n$, $\epsilon \in [0, 1]$
Output : $(1 - \epsilon)$ -MMS Allocation

```

1  $\mathbb{A} \leftarrow \emptyset$ .  $\Pi_n(\text{Big}) \leftarrow$  all partitions of Big into  $n$  sets.
2 for  $B \in \Pi_n(\text{Big})$  do
3   Define  $\mathcal{B}$ -sets as per equation (5.3).
4   while  $\mathcal{B}_1 \neq \emptyset$  and  $\text{Small}^- \neq \emptyset$  do
5     Remove any item  $j$  from  $\text{Small}^-$  and assign to any agent with a  $\mathcal{B}_1$  bundle
6     Re-define the  $\mathcal{B}$ -sets for the new allocation
7   if  $\mathcal{B}_1 = \emptyset$  then
8      $A \leftarrow \text{Bag-Fill}((n, M, v), \mathcal{B}\text{-sets})$ 
9     return  $A$ 
10   $\mathcal{A}^{1,2} \leftarrow$  allocation of all bundles from  $\mathcal{B}_1$  and  $\mathcal{B}_2$  to distinct agents
11   $n' \leftarrow |\mathcal{B}_3| + |\mathcal{B}_4|$ ,  $M' \leftarrow \cup_{B \in \mathcal{B}_3 \cup \mathcal{B}_4} B \cup \text{Small}^+$ 
12  if  $\mathcal{B}_4 \neq \emptyset$  then
13    if  $v(M') \geq n'(1 - \frac{\epsilon}{2})$  then
14       $A \leftarrow \mathcal{A}^{1,2} \cup \text{Bag-Fill}((n', M', v), \mathcal{B}\text{-sets} = \mathcal{B}_3 \cup \mathcal{B}_4)$ 
15      return  $A$ 
16    else
17      continue //  $B$  is invalid
18  else
19     $M' \leftarrow \text{Small}^+$ 
20    for  $B \in \mathcal{B}_3$  do
21      introduce a new good  $b$  with  $v(b) = v(B)$ ;  $M' \leftarrow M' \cup \{b\}$ 
22     $A \leftarrow \mathcal{A}^{1,2} \cup (1 - \epsilon)\text{-MMS allocation for } (n', M', v) \text{ using the algorithm in [34]}$ 
23     $\mathbb{A} \leftarrow \mathbb{A} \cup \{A\}$ 
24 return  $\text{argmax}_{A \in \mathbb{A}} \min_{A_i \in A} v(A_i)$ 

```

Proof. We form an allocation of $\cup_{B \in \mathcal{B}_3} B \cup \text{Small}^+$ among $n - |\mathcal{B}_1| - |\mathcal{B}_2|$ agents with the smallest bundle's value at least $\mu^n(M)$, thus proving the lemma. Consider the allocation of Small in A^* . Allocate the items from $M' = \cup_{B \in \mathcal{B}_3} B \cup \text{Small}^+$ among the set n' remaining agents who have received bundles in $\mathcal{B}_3$, as they are allocated in A^* . Call this allocation A' . Now the allocation A^* may also have some Small chores assigned to n' agents, but no other goods. The lowest valued bundle in A' thus has value at least that of the lowest valued bundle in A^* (since no SMALL chore is added to these bundles in A'). The MMS value of the remaining n' agent, when partitioning M' among them, is at least that of the lowest valued bundle of A' , hence is at least $\mu^n(M)$. $\square$

Next we state and prove the main theorem of this section.

Theorem 5.10. *Given an instance $\mathcal{I} = \langle n, M, v \rangle$ with $\mu \geq 0$, Algorithm 14 returns a $(1 - \epsilon)$ -MMS allocation in $O(m)$ time.*

Proof. First we prove the correctness of the algorithm. Note that no valid partition is discarded, as the procedure before deciding to discard a partition is exactly the procedure to determine if the partition is invalid. Consider a valid partition B^* corresponding to an MMS allocation A^* , and its $\mathcal{B}$ -sets as per (5.3). From Lemma 5.7, such a partition exists. After executing the while loop on Line 4, as every Small chore has absolute value at most $\epsilon/2$, upon adding the last chore before the value falls below 1, the value of every bundle

to which a chore was added is still at least $1 - \epsilon/2$. After this, one of the cases based on which conditions from $\mathcal{B}_1 = \emptyset$ and $\mathcal{B}_4 = \emptyset$ are true gets executed. In every case, there is some allocation generated, as the partition is valid.

If the Bag-Fill algorithm is called, then every agent gets a bundle of value $1 - \epsilon$. As $\mu \leq 1$, the allocation returned is $(1 - \epsilon)$ -MMS.

If the PTAS of [34] is called, then first, the agents receiving bundles from $\mathcal{B}_1, \mathcal{B}_2$, by definition of these sets, have value at least $1 - \epsilon \geq (1 - \epsilon)$ -MMS for their bundle. Also, as B^* corresponds to an MMS allocation, the MMS value for allocating the remaining items among the remaining agents, from Lemma 5.9, is at least the original MMS value. Hence, a $(1 - \epsilon)$ -MMS allocation of the goods manna instance, combined with the allocations to the agents with the $\mathcal{B}_1$ and $\mathcal{B}_2$ bundles, is $(1 - \epsilon)$ -MMS.

As B^* is considered when enumerating all the Big item partitions, this allocation will be stored in $\mathbb{A}$. Hence, the allocation returned has value at least $(1 - \epsilon)$ -MMS for the smallest valued bundle.

For running time, note that every iteration of the for loop first allocates all **Small** chores, then either runs a bag-filling algorithm which takes $O(m)$ time, discards the iteration, or runs the PTAS of [34] which takes $O(2^{\tilde{O}(1/\epsilon)} n \log m) = o(2^{(1/\epsilon^2)} n \log m)$ time. In the worst case, every iteration takes $O(m + O(2^{1/\epsilon^2} n \log m))$ time. The for loop runs for $n^{|\text{BIG}|}$ iterations, which from Lemma 5.6 is $O(n^{n/\tau\epsilon}) = 2^{O(n \log n/\tau\epsilon)}$. Hence, the total run time of the algorithm is $O(2^{n \log n/\tau\epsilon} (2^{1/\epsilon^2} n \log m + m)) = O(m)$ time. $\square$

5.4 Finding MMS values of Agents When $\mu < 0$

In this section we introduce the algorithm that finds a $(1 - \epsilon)$ -MMS allocation of an agent with $\mu < 0$ for an instance $\mathcal{I} = \langle n, M, v \rangle$ and a constant $\epsilon > 0$, or equivalently, a $(1 - \epsilon)$ -MMS allocation of $(N, M, (v_i)_{i \in N})$ when there are identical agents with valuation function v (Algorithm 15). From Lemmas 2.4 and the normalization $v(M) = -n$, we have $\mu \leq -1$.

From Definition 5.2, a $(1 - \epsilon)$ -MMS allocation gives each agent a bundle with value at least $(1/(1 - \epsilon))\mu$. Let $\sigma := \frac{1}{1 - \epsilon} - 1$. Algorithm 15 obtains an allocation where each agent gets a bundle of value at least $(1 + \sigma)\mu = (1/(1 - \epsilon))\mu$. The high level idea of the algorithm is as follows. First we scale the valuations so that $v(M) = -n$, and classify items as **Big** or **Small**. Then similarly as in Algorithm 14, we enumerate all partitions of **Big**. While there are unallocated **Small** goods, we add them one by one to the bundle with the least value. Once all the **Small** goods are exhausted, we iteratively add **Small** chores to the bundle with the highest value.

Theorem 5.11. *Algorithm 15 gives a $(1 - \epsilon)$ -MMS allocation when $\mu < 0$ in $O(m)$ time.*

Proof. We first prove a helpful lower bound on the value of all **Small** goods. Let B^* be a partition of the **Big** items corresponding to an MMS allocation. There are enough **Small** goods to add to each part in B^* so that every part has at least MMS value. Specifically, for the set $\mathcal{S} = \{B \in B^* : v(B) < \mu\}$ we have,

$$v(\text{Small}^+) \geq \mu \cdot |\mathcal{S}| - v(B^*) \quad (5.6)$$

Now, let $A = \{A_1, \dots, A_n\}$ be the output of Algorithm 15. Suppose for contradiction there exists some $A_i \in A$ such that $v(A_i) < (\frac{1}{1 - \epsilon})\mu = (1 + \sigma)\mu$. Consider each $A_k \supseteq B_k$ with $B_k \in \mathcal{S}$. Note that the algorithm adds **Small** goods to the bundle with the least value. Because of A_i , before adding the last **Small** good to any bundle, its value is less than $(1 + \sigma)\mu$. The last good added has value at most σ . Therefore, all the A_k s have

Algorithm 15: $(1 - \epsilon)$ -MMS Allocation for identical agents with $\mu < 0$

Input : $\mathcal{I} = \langle n, M, v \rangle$, a constant ϵ
Output : $(1 - \epsilon)$ -MMS Allocation

```

1 Normalize the valuations so that  $v(M) = -n$ .
2  $\sigma \leftarrow \frac{1}{1-\epsilon} - 1$ ,  $\mathbb{A} \leftarrow \emptyset$ 
3  $\text{Big} := \{j \in M : |v_j| \geq \sigma\}$ ,  $\text{Small} := M \setminus \text{Big}$ ,  $\text{Small}^+ = \text{Small} \cap M^+$ ,  $\text{Small}^- = \text{Small} \cap M^-$ 
4 for  $\mathcal{B} \in \Pi_n(\text{Big})$  do
5   while  $\text{Small}^+ \neq \emptyset$  do
6      $\perp$  add any Small good to a bundle with the lowest value
7   while  $M \neq \emptyset$  do
8      $\perp$  add any Small chore to a bundle with the highest value
9    $\perp$  store the allocation to a set  $\mathbb{A}$ 
10 return  $\text{argmax}_{A \in \mathbb{A}} \min_{A_i \in A} v(A_i)$  // return allocation with highest maximin value
  
```

value at most MMS. From, (5.6) and the fact that the algorithm adds goods to the least valued bundle, we have,

$$v(\text{Small}^+ \setminus (\bigcup_{k \in [n]} A_k)) \geq \sum_{B \in \mathcal{S}} \mu - v(B) + \sigma = \sigma, \quad (5.7)$$

which is a contradiction.

Now we prove that after adding the **Small** chores the value of all the bundles is at least $(1 + \sigma)\mu$. This is true because while there exists an unallocated chore, the value of the highest valued bundle is greater than -1 , because $v(M) = -n$. Adding a chore to such bundle will decrease the value by at most σ . Therefore, the value of such bundle is at least $-(1 + \sigma) \geq (1 + \sigma)\mu$. By definition of σ , $(1 + \sigma) = 1/(1 - \epsilon)$.

Finally, $|\text{BIG}| = O(n/\sigma) = O(n/\epsilon)$, from the definition of **BIG** and σ . As every iteration corresponding to a partition of **BIG** takes $O(m)$ time, Algorithm 15 runs for $O(m \cdot 2^{O(n \log n/\epsilon)}) = O(m)$ time. $\square$

5.5 Conclusions

In this chapter, we studied fair allocation of indivisible items in mixed manna under the notion of maximin share (MMS). In the mixed manna, finding even an approximate MMS value of an agent up to any approximation factor in $(0, 1]$ is NP-Hard for general instances [38]. In this chapter, we complemented the hardness result by obtaining a PTAS to compute the MMS value, when the following two conditions hold (i) number of agents n is a constant, and (ii) for every agent i , her total (absolute) value for all the items ($|v_i(M)|$) is significantly greater than the minimum of her total value of goods (v_i^+) and her total (absolute) value for chores (v_i^-), i.e., for a constant $\tau > 0$, $|v_i(M)| \geq \tau \cdot \min\{v_i^+, v_i^-\}$.

Chapter 6

Bounded MMS Value in Mixed Manna

6.1 Introduction

In the previous chapter we studied the problem of finding a PTAS for MMS value of agents in the mixed manna setting. We studied a restricted setting because of the result from [38] where they proved it is NP-hard to approximate MMS values within any constant factor. Their reduction indicates that perhaps the bottleneck issue that makes the problem hard, is that the absolute MMS value can be arbitrarily small. Intuitively speaking, as the MMS value approaches arbitrarily close to zero, the problem of finding approximate MMS values approaches that of finding exact MMS values. In the limit where MMS value is 0, every approximate MMS value is the exact MMS value. We then ask,

Q: If we had a guarantee that the absolute MMS value is greater than some threshold value, say δ , then does the problem become tractable?

In this chapter, we resolve this question positively for a specific value of δ . Note that since the MMS problem is scale-free, setting δ to a fixed constant will not make the problem any easier. Therefore, δ will have to be instance dependent. To be specific, let v^+ denote the sum of values of an agent for all the items she values positively, and v^- the sum of absolute values of all her negatively valued items (chores).

Theorem 6.1 (Informal). *In the case of identical agents, let μ be the MMS value, then there is an algorithm that: (a) when $|\mu| \geq \min\{v^+, v^-\}/\rho$ for some constant $\rho \geq 1$, finds an allocation that gives every agent a bundle of value at least $(1 - \epsilon)\mu$ for any constant $\epsilon > 0$, and (b) when $|\mu| < \min\{v^+, v^-\}/\rho$, reports this by returning the trivial allocation where all items are given to one agent. The algorithm runs in time $O(mnL)$, where m, n are the number of items and agents, and L is the bit-length of the input.*

We note that our assumption is weaker than having $\min\{v^+, v^-\}$ being a constant. Also, the extensively studied good manna and chore manna are special cases of this setting, hence any algorithmic results here translate to these settings as well.

One of the key tools used by our algorithm is a carefully designed Integer Program (IP) that can be solved in polynomial-time. IPs have been used to solve related problems in several works. [48] gave a PTAS using this idea for the machine covering problem, which is equivalent to the MMS problem in a good manna. The MMS problem in a chore manna is equivalent to machine scheduling which has a PTAS using IP for bin packing (the dual problem). Several algorithms for the bin packing problem solve a relaxation of an IP as their main idea [25, 36, 37]. Our approach builds on these, but requires several new ideas to handle both goods and chores simultaneously. Next we briefly describe some of these.

Non-constant variables. The variables of the IP will correspond to subsets of items. We show that we only need to consider subsets with total value at most a particular bound. With a good (chore) manna, this restricts to subsets where the number of items with value at least some fraction of the bound is a constant. While for a mixed manna, subsets of even $O(m)$ size may have a small value due to positive and negative values may cancel each other. Hence, the number of variables of the IP is not a constant for a mixed manna.

We circumvent this issue by reducing the problem to a problem with only goods, where a restricted set of allocations are allowed, called *valid allocations*.

Allow only valid allocations. The next task is to define constraints in the IP that ensure a valid allocation. Towards this, we define a cost function that characterizes valid allocations with a single constraint.

Sign of MMS. Our approach works for both cases $\mu \geq 0$ (Section 6.3), and $\mu < 0$. These are inherently different problems. The $\mu \geq 0$ problem maximizes the smallest bundle's value, while the negative MMS case minimizes the absolute value of the largest bundle. We show that our IP for the former case can be modified to work for the later case of $\mu < 0$.

In this section, we formally define mixed instances and other relevant notions of maximin share. We use $[k]$ to denote the set $\{1, 2, \dots, k\}$, and $(\mathcal{S}_j)_{j \in [k]}$ to denote the (multi-)set $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k\}$.

Definition 6.2. An MMS instance is a tuple $\mathcal{I} = \langle n, M, v \rangle$, where N is a set of n agents, M is a set of m indivisible items, and $v : 2^M \rightarrow \mathbb{R}$ is the identical additive valuation function of all agents, represented by $v(S) = \sum_{j \in S} v_j$ for $S \subseteq M$.

A partition of all items among all agents is termed an *allocation*, denoted by $A^\pi = \{A_1, A_2, \dots, A_n\}$. Thus, $A_i \cap A_{i'} = \emptyset$ for all distinct i, i' in N , and $\cup_i A_i = M$.

Definition 6.3 (MMS value). Given an MMS instance, let $\Pi_n(M)$ be the set of all possible allocations of M into n sets. The maximin share (MMS) value of an agent, denoted by $\mu^n(M)$, is defined as,

$$\mu^n(M) = \max_{A^\pi \in \Pi_n(M)} \min_{A_k \in A^\pi} v(A_k) .$$

We refer to $\mu^n(M)$ by MMS when the qualifiers n and M are clear. Note that MMS can be negative too.

An allocation which gives every agent a set of items worth at least MMS is called an MMS allocation. Note that all agents have the same valuation function v for M , hence the MMS values are same for all agents. Also, the allocation determining the MMS value for any agent is an MMS allocation. Hence when the agents are identical MMS allocations always exist. However, finding an MMS allocation is known to be NP-Hard [20]. Thus, we search for a PTAS to find almost optimal allocations, termed α -MMS allocations, defined as follows.

Definition 6.4 (MMS problem). Given an MMS instance $\mathcal{I} = \langle n, M, v \rangle$, the MMS problem is to find a α -MMS allocation of M among N .

Items of a mixed manna can be divided into two sets. *Goods* are the items valued positively according to v . The set of goods is denoted by $M^+ = \{j \in M \mid v_j \geq 0\}$. *Chores* are items valued negatively, and the set of chores is termed $M^- = \{j \in M \mid v_j \leq 0\}$. We denote by v^+ the sum of values of all goods in the manna. That is, $v^+ = \sum_{j \in M^+} v_j$. Similarly, we denote by v^- the sum of absolute values of all chores, i.e., $v^- = \sum_{j \in M^-} |v_j|$.

To circumvent the hardness result of [38] for the MMS problem for any $\epsilon \in [0, 1)$, we make the assumption that $|\mu| \geq \min\{v^+, v^-\}/\rho$, for some constant $\rho \geq 1$. As we cannot decide before computing the MMS value if a given instance satisfies this property, we pose the following problem, termed the Bounded MMS problem, denoted by B-MMS.

Definition 6.5 (B-MMS problem). *Given an MMS instance $\mathcal{I} = \langle n, M, v \rangle$ and an $\epsilon > 0$, return a α -MMS allocation if $\mu \geq \min\{v^+, v^-\}/\rho$ for some constant $\rho \geq 1$, else report $\mu < \min\{v^+, v^-\}/\rho$ by returning the trivial allocation where one agent gets all items M .*

In this chapter, we give a polynomial time algorithm that solves the B-MMS problem. In other words, we provide a PTAS to find the MMS values of agents in a mixed manna, when the absolute MMS values are higher than $\min\{v^+, v^-\}/\rho$, for some constant $\rho \geq 1$.

The following lemma by [38] shows an easy way to decide the sign of μ , allowing us to design separate approaches for the negative and non-negative MMS cases. (Lemma 5.5)

For solving the B-MMS problem, we first find the sign of MMS using Lemma 5.5, then apply the appropriate algorithm for that case.

6.2 Preliminaries and Notation

In Chapter 2, we defined the MMS value and MMS partition of an agents (Definition 2.1), as well as the MMS allocation, α -MMS allocation, and the MMS problem (Definition 2.2) for the instance $\mathcal{I} = \langle N, M, V \rangle$ with $|N| = n$. To avoid repetition, refer to Chapter 5 Section 5.2 for the notations and definitions of identical agents in mixed instances. We use $(\mathcal{S}_j)_{j \in [k]}$ to denote the (multi-)set $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k\}$.

In this Chapter, to circumvent the hardness result of [38] for the MMS problem for any $\epsilon \in [0, 1]$, we make the assumption that $|\mu| \geq \min\{v^+, v^-\}/\rho$, for some constant $\rho \geq 1$. As we cannot decide before computing the MMS value if a given instance satisfies this property, we pose the following problem, termed the Bounded MMS problem, denoted by B-MMS.

Definition 6.6 (B-MMS problem). *Given an MMS instance $\mathcal{I} = \langle n, M, v \rangle$ and an $\epsilon > 0$, return a α -MMS allocation if $\mu \geq \min\{v^+, v^-\}/\rho$ for some constant $\rho \geq 1$, else report $\mu < \min\{v^+, v^-\}/\rho$ by returning the trivial allocation where one agent gets all items M .*

In this chapter, we give a polynomial time algorithm that solves the B-MMS problem. In other words, we provide a PTAS to find the MMS values of agents in a mixed manna, when the absolute MMS values are higher than $\min\{v^+, v^-\}/\rho$, for some constant $\rho \geq 1$.

Similar to Chapter 5, Lemma 5.5 is used to decide the sign of μ , allowing us to design separate approaches for the negative and non-negative MMS cases.

For solving the B-MMS problem, we first find the sign of MMS using Lemma 5.5, then apply the appropriate algorithm for that case.

6.3 Algorithm for B-MMS when $\mu \geq 0$

In this section, we describe a PTAS for the B-MMS problem for the $\mu \geq 0$ case. The main parts of the PTAS are explained and solved in separate subsections.

6.3.1 Reducing B-MMS to MMS-like

We first reduce the given B-MMS problem to a new problem with only goods called the Goods manna Constrained MMS problem, denoted by MMS-like. At a high level, this is similar to the MMS problem, but it computes optimal allocations over a restricted set of partitions, called *valid allocations* (described shortly).

The intuition behind defining MMS-like problem is as follows: Suppose we replace every chore by $n - 1$ goods, each of value equal to the absolute value of the chore. Lets call these good-copies of the chores. Every time we want to assign a chore $j \in M$ to some agent, we instead assign one of the $n - 1$ good-copies of j to the remaining $n - 1$ agents (one copy for each $n - 1$ agents). This adds exactly $|v_j|$ value to every bundle and therefore keeps their relative order the same. Once we do this for every chore, the value added to each bundle is exactly v^- , and is the same for every partition in $\Pi^n(M)$. Therefore, if we restrict the allocations in the new setting to allow an agent to get at most one good-copy of any chore, then MMS allocations in the two settings are equivalent.

Following this intuition, we define a MMS-like instance and valid allocations as follows.

Definition 6.7 (MMS-like instance). *A tuple $\langle N, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle$, where N is a set of agents, G is a set of goods, $(\mathcal{S}_j)_{j \in [m^-]}$ are m^- sets of goods, each containing $(n - 1)$ identical copies of a good, and $u : M \cup (\mathcal{S}_j)_{j \in [m^-]} \rightarrow \mathbb{R}_+$ is the identical valuation function of the agents in N for all items $\mathcal{G} \cup (\mathcal{S}_j)_{j \in [m^-]}$.*

Definition 6.8 (Valid allocation). *Given a MMS-like instance $\langle n, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle$, an allocation A^π is valid if no agent receives more than 1 item from any set $\mathcal{S}_j \in (\mathcal{S}_j)_{j \in [m^-]}$, i.e., for all $i \in N, j \in [m^-]$, $|A^\pi_i \cap \mathcal{S}_j| \leq 1$.*

The MMS-like problem asks to find a *valid allocation* that maximizes the value of the smallest bundle, i.e., an MMS allocation over the valid allocations. We abuse notation to denote both the problem and the value by MMS-like, and formally define them as follows.

Definition 6.9 (MMS-like value). *Given a MMS-like instance $\langle n, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle$, let $\mathcal{F}$ be the set of all valid allocations. The MMS-like value of the instance, denoted by MMS-like, is defined as follows.*

$$\text{MMS-like} = \operatorname{argmax}_{A \in \mathcal{F}} \min_{A \in \mathcal{A}} v(A)$$

Since it is NP-hard to compute MMS-like (even when $(\mathcal{S}_j)_{j \in [m^-]} = \emptyset$), define the following approximate version of the problem.

Definition 6.10 (MMS-like problem). *Given a MMS-like instance $\langle n, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle$ and $\epsilon > 0$, return a valid allocation A^π such that $\min_{A \in A^\pi} v(A) \geq \alpha$ -MMS-like.*

Next we show that the B-MMS problem can be reduced to MMS-like problem such that a PTAS for the latter gives a PTAS for the former.

Given an instance $\mathcal{I} = \langle n, M, v \rangle$ we define the corresponding MMS-like instance $\langle n, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle$ as follows: The set of agents is unchanged, $\mathcal{G} = M^+$, $m^- = |M^-|$, and for all $j \in M^-$, define $\mathcal{S}_j$ to be a set of $(n - 1)$ goods represented as $\mathcal{S}_j := \{(j, k) | k \in [n - 1]\}$ – $\mathcal{S}_j$ consists of good-copies of chore j . Finally, define $u(j) = v(j)$ for all $j \in G$ and $u((j, k)) = -v(j)$ for all $j \in M^-$ and $k \in [n - 1]$.

Lemma 6.11. *Allocations of B-MMS are in one-to-one correspondence with valid allocations of MMS-like, such that if allocation B^π of the former maps to allocation C^π of the later then $u(C_i) = v(B_i) + v^-$, $\forall i \in N$.*

Proof. Given a B-MMS allocation B^π , add good-copies of each chore to agents who did not receive the chore in B^π , and discard all chores. This gives a MMS-like allocation C^π . The reverse allocation is obtained by similarly discarding all good-copies and assigning the corresponding chore to the agent who did not receive any good-copy.

Every agent $i \in N$ receives in C_i all the goods assigned to her in B_i . Every chore that was assigned to her in B_i is discarded in C_i . Due to this, her value increases by the absolute value of chores allotted

to her in B_i . Further, for every chore not assigned to her, she receives a good-copy of it in C_i . As for all $j \in C$, $u(j) = |v(j^*)|$ for the corresponding j^* in M , each good-copy increases her value by the absolute value of the corresponding chore. Her total valuation increases by the absolute value of all chores not assigned to her as well. Hence, the difference $u(C_i) - v(B_i)$ is exactly, $\sum_{j \in B_i} v(j) + \sum_{j \notin B_i} v(j) = v^-$. $\square$

Corollary 6.12. *MMS-like, relates to the MMS value of the B-MMS problem as,*

$$\text{MMS-like} = \mu + v^-. \quad (6.1)$$

Equation (6.1) allows to relate the approximation parameters of B-MMS and MMS-like allocations as follows.

Theorem 6.13. *If $\mu \geq v^-/\rho$, then a $(1 - \frac{\epsilon}{(1+\rho)})$ MMS-like allocation gives a α -MMS allocation, and therefore a PTAS for MMS-like gives a PTAS for the B-MMS problem.*

Proof. Let $\epsilon' = \frac{\epsilon}{(1+\rho)}$. Take the $(1 - \epsilon')$ MMS-like allocation, say C^π , and consider the corresponding allocation B^π of the B-MMS instance as described in the proof of Lemma 6.11. From Lemma 6.13, the smallest bundle in B^π has value $(1 - \epsilon')\text{MMS-like} - v^-$.

If $\mu \geq v^-/\rho$, we have, $(1 - \epsilon')\text{MMS-like} - v^- \geq (1 - \epsilon')(\mu + v^-) - v^- \geq (1 - \epsilon')(\mu + \rho\mu) - \rho\mu = (1 - (1 + \rho)\epsilon')\mu = (1 - \epsilon)\mu$. Therefore, B^π is a α -MMS allocation

Since ρ and ϵ are constants in the B-MMS problem, ϵ' is also a constant. Therefore, a PTAS for MMS-like is indeed a PTAS for the B-MMS problem as well. $\square$

Due to the above theorem, it suffices to obtain a PTAS for the MMS-like problem.

6.3.2 Algorithm for MMS-like

Algorithm 16 for MMS-like will perform a search for the highest value Δ for which we get an allocation that gives every agent at least a Δ -valued bundle. For this we perform a search on a multiplicative grid over all possible values of MMS-like, obtained as follows. First, we have $v^-/\rho \leq \mu \leq v(M)/n = (v^+ - v^-)/n$. Combined with Equation (6.1), we get $v^- + v^-/\rho \leq \text{MMS-like} \leq (v^+ - v^-)/n + v^-$.

In each iteration of the search, it first checks if there is an item with value more than Δ . First, there will be no such chore. Because if there was one, say c , we have $c > \Delta \geq \text{MMS-like} = \mu + v^-$ which implies $\mu < c - v^- \leq 0$.

If there is a good j with $v(j) \geq \Delta$, we have $\Delta - v^- \geq \text{MMS-like} - v^- = \mu(\text{B-MMS instance})$. Using this we find B^π , a solution of the B-MMS instance as follows: assign good j and all the chores to an agent, and remove the agent and her bundle. The following allocation for the resulting instance is feasible, and has equal or higher MMS value. From any MMS allocation of the B-MMS instance, (a) remove all chores and add them to the part containing the good j , and (b) remove all goods except j from this part and arbitrarily distribute among the remaining parts. The MMS value of the resulting instance is not lower, and therefore it suffices to find its α -MMS allocation using the PTAS of [48]. Algorithm 16 returns allocation C^π corresponding to this B-MMS allocation.

If every item has value at most Δ , the algorithm applies a subroutine Exists-GC-MMS, for which we prove in Section 6.3.3,

Theorem 6.14. *Exists-GC-MMS($\langle n, G, (\mathcal{S}_j)_{j \in [m-]}, u \rangle, \epsilon, \Delta$) returns a tuple (A^π, flag) with $\text{flag} = \text{true}$ and $u(A) \geq (1 - \epsilon)\Delta$, $\forall A \in A^\pi$, whenever $\Delta \leq \text{MMS-like}$. And it runs in $O(mn)$ time.*

Algorithm 16: Algorithm for MMS-like

Input : $\langle n, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle, \epsilon' > 0$
Output : α -MMS-like allocation if $\text{MMS-like} \geq (1 + 1/\rho)v^-$

```
1  $\bar{\epsilon} \leftarrow \epsilon'/2; \Delta \leftarrow v^+/n + (1 - 1/n)v^-$ 
2 while  $\Delta \geq (1 + 1/\rho)v^-$  do
3   if  $\exists j \in \mathcal{G} : u(j) \geq \Delta$  then
4      $A^\pi = (A_1, \dots, A_n), A_n \leftarrow \{j\}$ 
5      $(A_1, \dots, A_{n-1}) \leftarrow (1 - \bar{\epsilon})\text{-MMS partition of } \langle N \setminus \{n\}, G, u \rangle$  // use PTAS of [48]
6      $A_i \leftarrow A_i \cup \{(j, i)\}$  for all  $i \in [n - 1]$  and  $j \in [m^-]$ 
7     return  $A^\pi$ 
8    $(A^\pi, \text{flag}) \leftarrow \text{Exists-GC-MMS}(\langle n, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle, \bar{\epsilon}, \Delta)$ 
9   If  $\text{flag}$  then return  $A^\pi$ 
10  else  $\Delta \leftarrow (1 - \bar{\epsilon})\Delta$ 
11  $A^\pi = (A_1, \dots, A_n)$  where  $A_i = \{(j, i) : \forall j \in [m^-]\}$  for  $i \in [n - 1], A_n = G$  // agents 1 to  $n - 1$ 
    each get one good-copy of all chores.
12 return  $A^\pi$ 
```

If Exists-GC-MMS returns a false flag, the Algorithm resets $\Delta \leftarrow (1 - \epsilon)\Delta$ and starts the next iteration, else returns the allocation obtained and stops. Theorem 6.14 implies the following. When Algorithm 16 stops, say for a value Δ^* , we know $\text{MMS-like} \leq \Delta^*/(1 - \bar{\epsilon})$, from the false flag returned in the previous iteration. From this iteration's output, we have a $(1 - \bar{\epsilon})^2 \text{MMS-like}$ allocation. Fixing $\bar{\epsilon}$ as $\epsilon'/2$ gives,

Lemma 6.15. *If $\text{MMS-like} \geq (1 + 1/\rho)v^-$, Algorithm 16 returns a α -MMS-like allocation.*

Proof. First, suppose there is a good $j \in \mathcal{G} : u(j) \geq \Delta$, the B-MMS allocation corresponding to the MMS-like allocation returned gives the single good with all chores to one agent, say i . The value of i 's B-MMS bundle is $\Delta - v^-$. Note that in every iteration of the algorithm, $\Delta \geq \text{MMS-like}$, as the algorithm stops when it finds the first Δ which gives every agent a bundle of value at least Δ . Hence, from Equation (6.1), the value of i 's B-MMS bundle is $\Delta - v^- \geq \text{MMS-like} - v^- \geq \mu$.

By allocating all chores to i , and a single good, the MMS values of the remaining agents over the remaining goods only increases. Hence, a $(1 - \epsilon')$ -MMS allocation of these, combined with i 's bundle, is a α -MMS-like allocation.

Now consider the second case when for all items $j \in \mathcal{G} : u(j) < \Delta$. Denote the value of Δ considered by Algorithm 16 for which Exists-GC-MMS returns $\text{flag} = \text{true}$ by Δ^* . From Theorem 6.14, the corresponding allocation A^π returned by Exists-GC-MMS has all bundles of value at least $(1 - \bar{\epsilon})\Delta^*$.

If Exists-GC-MMS returns $\text{flag} = \text{true}$ for the first value of Δ considered, then as this value is the upper bound on Δ , we have $\text{MMS-like} \leq \Delta^*$. Hence $(1 - \bar{\epsilon})\Delta^* \geq (1 - \bar{\epsilon})\text{MMS-like} \geq (1 - \epsilon')\text{MMS-like}$, where the last inequality follows as $\bar{\epsilon} = \epsilon'/2$.

Otherwise, for the value of Δ considered in the previous iteration before Δ^* , that is, for $\Delta^*/(1 - \bar{\epsilon})$, Exists-GC-MMS returned $\text{flag} = \text{false}$. Hence $\text{MMS-like} < \Delta^*/(1 - \bar{\epsilon})$. Hence, the smallest valued bundle in A^π has value at least $(1 - \bar{\epsilon})\Delta^* \geq (1 - \bar{\epsilon})^2 \text{MMS-like} \geq (1 - \epsilon')\text{MMS-like}$.

Thus, in both cases, A^π is a α -MMS-like allocation. $\square$

Now we are ready to show Theorem 6.1 for the case when $\mu \geq 0$.

Theorem 6.16. *There is an algorithm to solve the B-MMS problem for the case $\mu \geq 0$, that runs in time $O(mnL)$, where L is the number of bits needed to represent function v .*

Proof. From Theorem 6.13, it suffices to get a PTAS for the corresponding MMS-like problem. By Lemma 6.15 Algorithm 16 does solve a MMS-like problem. The while loop of the algorithm runs for $\frac{1}{\bar{\epsilon}} \log \left(\frac{v^+ + (n-1)v^-}{n} - \frac{(1+\rho)v^-}{\rho} \right) \leq \frac{2(1+\rho)}{\epsilon} L$ many times. By Theorem 6.14 and [48], every iteration of the while loop takes at most $O(mn)$ time, and therefore the overall running time is $O(mnL)$. $\square$

Remark 6.17. *The proof of Theorem 6.1 for the case when $\mu < 0$ is similar, and in some sense simpler, and discussed in Appendix 6.4.*

The next section shows Theorem 6.14.

6.3.3 Algorithm for Exists-GC-MMS

At a high level, we first map the set of items in the Exists-GC-MMS instance to multi-sets of numbers corresponding to their values (scaled to have $\Delta = 9\lceil \frac{1}{\epsilon} \rceil^2$ for technical reasons). *Valid partitions* of these numbers are defined analogously like valid allocations of the MMS-like items. We then classify the values as BIG or SMALL. The key component of the algorithm is an IP to find a valid partition of the BIG values such that (a) every part has value at least $9(\lceil \frac{1}{\epsilon} \rceil^2 - \lceil \frac{1}{\epsilon} \rceil)$, and (b) there are enough SMALL values to greedily allocate over this partition and have every part valued at least $9\lceil \frac{1}{\epsilon} \rceil^2$. We now discuss the details of the algorithm formally.

Exists-GC-MMS has two steps 1) Pre-processing and 2) Main Algorithm.

Pre-processing. (Algorithm 17, line 1) Let $E := \lceil \frac{1}{\epsilon} \rceil$. Note that E is a constant integer that only depends on $\bar{\epsilon}$ and not on parameters in the MMS-like instance. Scale the valuations v by $9E^2/\Delta$. Let $\mathcal{V}^g = (g_j)_{(j \in [m^+])}$ and $\mathcal{V}^c = \cup_{j \in [m^-]} C_j$, where $C_j = (c_j^k)_{k \in [n-1]}$ be multi-sets of numbers corresponding to scaled valuations, respectively of M^+ and $(\mathcal{S}_j)_{j \in [m^-]}$. Let $\mathcal{T} = \mathcal{V}^g \cup \mathcal{V}^c$.

This completes the pre-processing step. The following lemmas characterize partitions of $\mathcal{T}$ that correspond to approximately optimal MMS-like allocations.

Definition 6.18 (Valid Partition of $\mathcal{T}$). *We call a partition $P = (P_1, \dots, P_n)$ of values in $\mathcal{T}$ valid if each P_k contains at most one element from each C_j , i.e., $|P_k \cap C_j| \leq 1$ for all $k \in [n]$ and $j \in [m^-]$.*

It is easy to see that each valid partition of $\mathcal{T}$ is equivalent to a valid allocation in its corresponding MMS-like instance. With the scaling step, this directly implies,

Lemma 6.19. *Given a MMS-like instance $\langle n, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle$, if $\Delta \leq \text{MMS-like}$ then there is a valid partition of $\mathcal{T}$ where the sum of values in each part is at least $9E^2$.*

Proof. Since $\Delta \leq \text{MMS-like}$, there is a valid allocation for $\langle n, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle$ where all bundles have value at least Δ . If we scale the value of each item by $9E^2/\Delta$, the value of all bundles will also be scaled by $9E^2/\Delta$ because v is additive. Hence we get a valid partition where all parts have a total sum of at least $9E^2$. $\square$

As $E = \lceil 1/\bar{\epsilon} \rceil$, we can show that a part of value at least $9E^2 - 9E$ will correspond to a bundle of value at least $(1 - \bar{\epsilon})\Delta$. We use this and Lemma 6.19 to show the next lemma.

Lemma 6.20. *A valid partition of $\mathcal{T}$ where the sum of values in each part is at least $9(E^2 - E)$ is equivalent to a valid allocation for its corresponding MMS-like instance where each bundle has value at least $(1 - \bar{\epsilon})\Delta$.*

Proof. Let $P = (P_1, \dots, P_n)$ be the valid partition and let $A = (A_1, \dots, A_n)$ be its corresponding valid allocation. We have $\sum_{p \in P_i} p = (9E^2/\Delta)u(A_i)$. Hence,

$$\begin{aligned}
& \sum_{p \in P_i} p \geq 9(E^2 - E) & \forall i \in [n] \\
& \left(\frac{9E^2}{\Delta}\right)u(A_i) \geq 9(E^2 - E) & \forall i \in [n] \\
\Rightarrow & u(A_i) \geq \Delta \left(\frac{9E^2 - 9E}{9E^2}\right) & \forall i \in [n] \\
\Rightarrow & u(A_i) \geq \Delta \left(1 - \frac{1}{E}\right) & \forall i \in [n] \\
\Rightarrow & u(A_i) \geq (1 - \epsilon)\Delta & \forall i \in [n].
\end{aligned}$$

The last inequality holds because $E = \lceil \frac{1}{\epsilon} \rceil \geq \frac{1}{\epsilon}$. $\square$

Main Algorithm. Call a valid partition of $\mathcal{T}$ *optimal* if the sum of values in each part is at least $9(E^2 - E)$. This step returns an optimal partition if $\Delta \leq \text{MMS-like}$, else correctly reports $\Delta > \text{MMS-like}$ by returning $flag = false$. Note that Algorithm 1 runs **Exists-GC-MMS** only if every item has value at most Δ . Hence, after scaling by $9E^2$, we can assume $t \leq 9E^2$, $\forall t \in T$. The key of the algorithm is an IP. We first explain the IP.

Notation. We define **SMALL** and **BIG** values in $\mathcal{T}$. Call a value $t \in \mathcal{T}$ **SMALL** if $t < 3E$ and **BIG** if $t \geq 3E$. For each $T \subseteq \mathcal{T}$ let **SMALL**(T) be the set of all small values in T and **BIG**(T) be the set of all big values in T . We call a set $C_j \in \mathcal{V}^c$ *small* if it contains **SMALL** values and *big* otherwise¹. Let $\sigma, \sigma^+, (n-1) \cdot \sigma^-$ respectively be the sum of all values in **SMALL**($\mathcal{T}$), **SMALL**($\mathcal{V}^g$) and **SMALL**($\mathcal{V}^c$), i.e., $\sigma := \sum_{t \in \text{SMALL}(\mathcal{T})} t$, $\sigma^+ := \sum_{t \in \text{SMALL}(\mathcal{V}^g)} t$ and $\sigma^- := (\sum_{t \in \text{SMALL}(\mathcal{V}^c)} t)/(n-1)$. Note that σ^- is equal to the sum of values obtained by picking one value from each small C_j , and $\sigma = \sigma^+ + (n-1)\sigma^-$.

Next, we know that every **BIG** value will be in the range $[3E, 9E^2]$. For all integers r in $[3E, 9E^2]$, let n_r^+, n_r^- respectively be the number of values in **BIG**($\mathcal{V}^g$) and the number of sets C_j with integral part of values r . Thus, $(n-1)n_r^- + n_r^+$ items j in $\mathcal{V}^g \cup \mathcal{V}^c$ have $\lfloor j \rfloor = r$.

We now define notation to represent a subset of **BIG** values and their sum. Let X denote a part in a partition of $\mathcal{T}$. We define the *type* of X by $\tau(X) = \langle \underline{\tau}(X), \bar{\tau}(X) \rangle = (\underline{\tau}_{3E}, \dots, \underline{\tau}_{9E^2}, \bar{\tau}_{3E}, \dots, \bar{\tau}_{9E^2})$; here $\underline{\tau}_r, \bar{\tau}_r$ are resp. the number of values in **BIG**($X \cap \mathcal{V}^g$) and **BIG**($X \cap \mathcal{V}^c$) with integer part r . Let $\text{SIZE}(\tau(X)) := \sum_r r(\underline{\tau}_r + \bar{\tau}_r)$ be the total sum of these rounded values in **BIG**($\mathcal{T} \cap X$).

Using this notation, we design an IP to find an assignment of **BIG** values in an optimal partition. First, observe that every **BIG** value is at most $9E^2$. Thus, if an optimal partition has some part valued more than $18E^2$, we can remove values until the size of this set is in the range $[9E^2, 18E^2]$. Finding a *partial* allocation of **BIG** values that assigns at least $9E^2$ value to all parts suffices to solve **Exists-GC-MMS**, as we can arbitrarily add the unallocated values. Thus, we will only consider types whose size $\text{SIZE}(\cdot)$ is at most $18E^2$.

The variables of the IP correspond to all types τ that satisfy (i) $\text{SIZE}(\tau) \leq 18E^2$, (ii) $\underline{\tau}_r \leq n_r^+$, (iii) $\bar{\tau}_r \leq n_r^-$. Let $\tau^{(1)}, \tau^{(2)}, \dots, \tau^{(\Gamma)}$ be an enumeration of all variables. Intuitively, we consider types that represent valid allocations of items corresponding to the **BIG** values in a **MMS-like** instance. Every IP variable takes an integer value equal to the number of times the corresponding type is selected. This in turn represents

¹Note that each C_j , $j \in [m^-]$ contains $n-1$ equal values. i.e., for each C_j , either $\text{SMALL}(C_j) = \emptyset$ or $\text{SMALL}(C_j) = C_j$.

the number of parts in the output allocation that have a subset of BIG items as represented by this type.

Lemma 6.21. *The number of IP variables Γ is $O(1)$.*

Proof. Every type with size at most $18E^2$ can have at most $6E$ BIG values, as every BIG value is at least $3E$. Each value is one of $[3E, 9E^2]$, a constant sized set. Hence, the number of types $\bar{\tau}$ and $\underline{\tau}$ are each at most $(9E^2 - 3E + 1)^{6E}$. The total number of types at most twice this value, hence a constant as E is a constant. The number of variables of the IP is at most the number of types with size at most $18E^2$, hence is constant. $\square$

Before defining the IP, we define two *cost* functions for every type. These are used to define constraints to allocate **SMALL** items.

First, define $c(\tau(X)) := \max\{0, 9E^2 - 6E - \text{SIZE}(\tau(X))\}$. The intuition for this function is as follows. Our aim is to create an optimal partition. If the sum of BIG values $\text{SIZE}(\tau(X)) < 9(E^2 - E)$, we must add values from **SMALL**($\mathcal{T}$) to X . The required sum from **SMALL**, is at least $9E^2 - 9E - \text{SIZE}(\tau(X))$. However, **SMALL**($\mathcal{T}$) does not have arbitrarily precise values. As every **SMALL** value is at most $3E$, we may have to add **SMALL** items until the net value of the part becomes $3E$ more than required, i.e., $9E^2 - 6E$. Hence the cost function $c(\tau(X))$ is defined as specified.

The second cost function captures the value that must be added to a part from **SMALL**($\mathcal{V}^g$). If a part has $c(\tau) > 0$, we can add at most value σ^- to the part from **SMALL**($\mathcal{V}^c$). Hence, the minimum value from **SMALL**($\mathcal{V}^g$) is $\sigma^+(\tau(X)) := \max\{0, c(\tau(X)) - \sigma^-\}$.

Using these notions, we define the following IP for finding an allocation of BIG values.

$$\sum_{j=1}^{\Gamma} x_j = n; \quad x_j \in \{0\} \cup \mathbb{N}, \forall j \in [\Gamma] \quad (6.2)$$

$$\sum_{j=1}^{\Gamma} \underline{\tau}_r^{(j)} x_j \leq n_r^+ \forall r \in [3E, 9E^2] \quad (6.3)$$

$$\sum_{j=1}^{\Gamma} \bar{\tau}_r^{(j)} x_j \leq (n-1)n_r^-, \forall r \in [3E, 9E^2] \quad (6.4)$$

$$(a) \sum_{j=1}^{\Gamma} c(\tau^{(j)}) x_j \leq \sigma; \quad (b) \sum_{j=1}^{\Gamma} \sigma^+(\tau^{(j)}) x_j \leq \sigma^+ \quad (6.5)$$

The **Exists-GC-MMS** algorithm is as follows. After applying the pre-processing step, it defines and solves the above IP. If the IP has a solution, then first it considers the items from the **MMS**-like instance that correspond to the BIG values in $\mathcal{T}$. The algorithm partitions all these items in n bundles by creating n subsets, with x_i subsets corresponding to type $\tau^{(i)}$. After this, it considers the subsets of BIG items that do not have total sum of values at least $9E^2 - 9E$. To each of these, it first adds the **SMALL** items corresponding to the small C_j subsets, by adding at most one item from each subset C_j , in any order. If upon adding these, the value of the set is still not $9E^2 - 9E$, it adds items corresponding to the **SMALL**($\mathcal{V}^g$) set, until the total sum of values is at least $9E^2 - 9E$. The algorithm returns the tuple (A^π, true) , where A^π is the allocation formed by this process. If the IP does not have a solution, it returns the tuple $(\emptyset, \text{flag} = \text{false})$.

Algorithm 17 formally describes **Exists-GC-MMS**. We now analyze the correctness of **Exists-GC-MMS**.

Lemma 6.22. *If $\Delta \leq \text{MMS-like}$, then IP has a solution.*

Algorithm 17: Exists-GC-MMS

Input : $\langle n, G, (\mathcal{S}_j)_{j \in [m^-]}, u \rangle, \bar{\epsilon}, \Delta$
Output : (A^π, True) if there exists a $(1 - \bar{\epsilon})$ -MMS-like allocation A^π and $(\emptyset, \text{False})$ otherwise

```

1  $\mathcal{V}^g \leftarrow \{g_1, \dots, g_{m^+}\}, g_j = u(j) \cdot \left(\frac{9E^2}{\Delta}\right), j \in G$   $\mathcal{V}^c \leftarrow \bigcup_{j \in [m^-]} C_j, C_j := \{c_j^1, \dots, c_j^{n-1}\},$ 
    $c_j^k = u(j, k), \forall (j, k) \in \mathcal{S}_j, \forall \mathcal{S}_j \in (\mathcal{S}_j)_{j \in [m^-]}; \mathcal{T} \leftarrow \mathcal{V}^g \cup \mathcal{V}^c$ 
2 if IP has a solution  $X$  for  $\mathcal{T}$  then
3    $j \leftarrow 1$ 
4   for all  $i : x_i \neq 0$  : do
5     Create  $x_i$  parts  $P_j$  to  $P_{j+x_i}$ 
6     Add BIG values to each  $P_k, k \in [j, j+x_i]$  as per  $\tau^{(i)}$  ;  $j \leftarrow j+x_i+1$ 
7   while  $\exists k : \sum_{j \in P_k} j < (9E^2 - 9E)$  do
8     while  $\sum_{j \in P_k} j < (9E^2 - 9E)$  do
9       If  $P_k \cap C_j = \emptyset$  for any  $j \in [m^-]$  then add one value from  $C_j$  to  $P_k$ 
10      else add any value from  $\text{SMALL}(\mathcal{V}^g)$  to  $P_k$ 
11   while there is an unallocated value  $k$  from  $C_j$  for any  $j \in [m^-]$  do
12     Add  $k$  to any  $P_i : P_i \cap C_j = \emptyset$ 
13   Add remaining unallocated values arbitrarily
14    $A^\pi \leftarrow$  allocation corresponding to  $P = (P_1, \dots, P_n)$  // use Lemma 6.19
15   return  $(A^\pi, \text{True})$ 
16 return  $(\emptyset, \text{False})$ 

```

Proof. As $\Delta \leq \text{MMS-like}$, from Lemma 6.19, there is a valid partition of $\mathcal{T}$ with sum of values of each part at least $9E^2$. Let this partition be T^{IP} . Let $\tau^i = \tau(T_i^{\text{IP}})$ be the type of each part, and $\tau^{\text{IP}} = [\tau^1 \dots, \tau^n]$, be the multi-set of types of all parts.

Constraints (6.2), (6.3) and (6.4) hold for τ^{IP} by definition of a valid partition. For any $\tau^i \in \tau^{\text{IP}}$ with $c(\tau^i) = 0$, we have $\sum_{t \in \text{SMALL}(T_i^{\text{IP}})} t \geq c(\tau^i) = 0$, and for any $\tau^i \in \tau^{\text{IP}}$ with $c(\tau^i) > 0$, we have $\sum_{t \in \text{SMALL}(T_i^{\text{IP}})} t \geq 9E^2 - \sum_{t \in \text{BIG}(T_i^{\text{IP}})} t \geq 9E^2 - 6E - \text{SIZE}(\tau^i) \geq c(\tau^i)$. The second inequality holds because the SIZE function rounds down all values, and there are at most $6E$ BIG values in each T_i^{IP} . By adding the above inequality for all $\tau^i \in \tau^{\text{IP}}$, we obtain (6.5) of the IP.

Since each T_i^{IP} is a subset of a valid part, its corresponding type τ^i has at most one value from each C_j . Therefore, for any $\tau^i \in \tau^{\text{IP}}$ with $\sigma^+(\tau^i) > 0$ we have, for $\sum_{t \in \text{SMALL}(\tau^i \cap \mathcal{V}^g)} t \geq c(\tau^i) - \sum_{t \in \text{SMALL}(\tau^i \cap \mathcal{V}^c)} t \geq c(\tau^i) - \sigma^- \geq \sigma^+(\tau^i)$. Moreover, for any $\tau^i \in \tau^{\text{IP}}$ with $\sigma^+(\tau^i) = 0$ we have, for $\sum_{t \in \text{SMALL}(\tau^i \cap \mathcal{V}^g)} t \geq \sigma^+(\tau^i) = 0$. By adding the above inequality for all $T_i^{\text{IP}} \in T^{\text{IP}}$ we get constraint (6.5). Thus, T^{IP} is a solution of the IP. $\square$

Lemma 6.23. *If the IP has a solution, then the allocation returned by Exists-GC-MMS is an allocation that gives every agent a bundle of value at least $(1 - \bar{\epsilon})\Delta$.*

Proof. Let τ^{sol} be the solution of the IP and P^{sol} be the partition of the values formed by Exists-GC-MMS after finding τ^{sol} . We show that each part of P^{sol} has value at least $(9E^2 - 9E)$. From Lemma 6.20, we get that in A^π , every agent gets a bundle of value at least $(1 - \bar{\epsilon})\Delta$.

After assigning BIG values to P_i as per the type τ^i , suppose there are parts with value less than $9E^2 - 9E$.

Consider any such part P . The algorithm first adds SMALL values from $\mathcal{V}^g$. As τ^{sol} satisfies constraint (6.5) (a) of the IP, then $c(\tau(P)) \leq \sigma^+$. That is, the value to add to P so that the sum of values in P is at least $9E^2 - 9E$ is at most the sum of all SMALL values. We first add values from $\text{SMALL}(\mathcal{V}^c)$. Suppose after

receiving one value from each set in $\mathcal{V}^c$, P still has value less than $(9E^2 - 9E)$. As τ^{sol} satisfies constraint (b) of (6.5) of the IP, the total cost from $\text{SMALL}(\mathcal{V}^c)$ for all parts together is at most σ^+ . As the cost function is monotonic with number of parts, the total cost from $\text{SMALL}(\mathcal{V}^c)$ for P also is at most $\text{SMALL}(\mathcal{V}^c)$. Hence, there are enough values in $\text{SMALL}(\mathcal{V}^g)$ to add to P_i to increase its value to at least $(9E^2 - 9E)$.

After adding values to P , its total value is at most $9E^2 - 6E$, as every item has value at most $3E$. Thus, the value added to it from SMALL values is at most $c(\tau(P))$. The total cost of the remaining parts is $\sum_{P' \neq P} c(\tau(P')) = \sum_{P \in P^{sol}} c(\tau(P)) - c(\tau(P)) \leq \sigma^+ - (\text{the sum of SMALL values assigned to } P)$, which is exactly the total value of unassigned SMALL values. Hence, constraint 6.5 (a) is satisfied for the smaller set $\tau^{sol} \setminus \tau(P)$. Similarly, we can show constraint 6.5 (b) also is satisfied. The initial constraints 6.2, 6.4 and 6.3 are satisfied for $\tau^{sol} \setminus \tau(P)$ by the validity of τ^{sol} . Hence $\tau^{sol} \setminus \tau(P)$ is a solution to the IP for the smaller case after removing P and its assigned values. By induction, we can assign values to every part until all parts are satisfied. Adding any unallocated values arbitrarily in Line 13 only increases the value of each bundle.

Hence, the partition P^{sol} obtained has every bundle of value at least $9E^2 - 9E$. From Lemma 6.20, the corresponding allocation A^π gives every agent a bundle of value at least $(1 - \bar{\epsilon})\Delta$. $\square$

Lemma 6.24. *Exists-GC-MMS runs in time $O(mn)$.*

Proof. The time to run Exists-GC-MMS is asymptotically equal to the time for constructing and solving the IP. Lenstra's algorithm [41] takes time exponential in the number of variables, $O(2^{1/\epsilon^2}) = O(2^{4/\epsilon'^2})$ here, and polynomial in the largest coefficient of any variable in all inequalities, $m^+ + (n-1)m^- = O(mn)$ here. Note that σ and σ^+ are at most $n \cdot 9E^2$. Hence, the IP requires $O(2^{1/\epsilon^2} mn) = O(mn)$ time. $\square$

Lemmas 6.22, 6.23 and 6.24 together prove Theorem 6.14.

Algorithm 18: $\alpha\text{-MMS}\langle n, G, C, \Delta, \epsilon \rangle$

1 **Define** for $\epsilon > 0$: $E \leftarrow \lceil \frac{1}{\epsilon} \rceil$; $\mathcal{A} \leftarrow \emptyset^n$
2 **return** $\text{MMS-Large-Item}\langle n, G, C, \Delta, \epsilon \rangle$ OR $\text{MMS-PTAS}\langle n, G, C, \Delta, E \rangle$

6.4 Algorithm for B-MMS when $\mu < 0$

In this section, we will give an overview of the algorithm for the B-MMS problem when $\mu < 0$. That is, v^+ is less than v^- , and $|\mu| \geq v^+/\rho$, for some constant $\rho \geq 1$. To obtain this result, we reduce the B-MMS problem to the CC-MMS problem, defined shortly. We prove that a PTAS for the CC-MMS problem implies a PTAS for the B-MMS problem. To complete the result, in the major part of this Section, we show a PTAS for the CC-MMS problem.

6.4.1 Technical Overview

In the entire discussion, we will consider the value of each chore as its absolute value, intuitively representing the *cost* of doing the chore.

The outline of the algorithm is the same as the counter case when $\mu \geq 0$. To avoid repetition, we will focus on highlighting the main ideas that differ from the approach used in Section 6.3. We first reduce B-MMS to a problem with only chores, denoted by CC-MMS. We substitute every good of B-MMS by a set of $(n-1)$ chores in CC-MMS, and call them chore-copies of the good. An optimal allocation here has the *lowest* value

for the *largest* bundle (call this value CC-MMS). We find via a search algorithm, that runs a subroutine Exists-CC-MMS in each iteration, the lowest value Δ for which there exists an allocation that gives every agent at most a Δ valued bundle.

Exists-CC-MMS (Algorithm 20), scales the values of items by $9E^2/\Delta$ and calls the set of all scaled valuations (of chores and chore-copies) $\mathcal{T}$. We prove in Lemma 6.38 that a *valid partition of $\mathcal{T}$* (defined in Definition 6.36), defined as one where the sum of values in each part is at most $9E^2 + 9E$, implies a $\Delta/(1 - \epsilon)$ valued valid allocation for the corresponding CC-MMS instance. Then, Exists-CC-MMS classifies the set of values in $\mathcal{T}$ as $\text{BIG}(\mathcal{T})$ or $\text{SMALL}(\mathcal{T})$, solves an IP to find a suitable allocation of values in $\text{BIG}(\mathcal{T})$, and allocates the $\text{SMALL}(\mathcal{T})$ values greedily upon the allocation of $\text{BIG}(\mathcal{T})$.

We then define *types* of subsets of $\text{BIG}(\mathcal{T})$ with a vector $\tau = [\bar{\tau}, \underline{\tau}]$. However, when $\mu < 0$, we consider the closest integer to the values when they are rounded *up*. The variables of the IP designed for Exists-CC-MMS correspond to all type vectors τ that represent all bundles of $\text{BIG}(\mathcal{T})$ with total value *at most* $9E^2 + 9E$, and those that contain at most one chore-copy of every BIG good.

The IP has constraints to select n types that ensure the selection *at least* covers all given chores and chore-copies. We now define two *surplus* functions. The first, denoted by $c(\tau(X))$, represents the cost that can be added to every bundle X from $\text{SMALL}(\mathcal{T})$ while keeping the bundle's value at most $9E^2 + 6E$. This highest allowed value is $3E$ lower than our desired bound of $9E^2 + 9E$. This is because we cannot form bundles of $\text{SMALL}(\mathcal{T})$ values of arbitrary precision. At a high level, if the IP finds a solution where all the $\text{SMALL}(\mathcal{T})$ values can be filled while keeping every bundle's total cost at most $9E^2 + 6E$, then as every $\text{SMALL}(\mathcal{T})$ value is at most $3E$, we can greedily allocate all these without any bundle's cost exceeding $9E^2 + 9E$. Formally, if $\text{Size}(\tau(X))$ is the cost of a bundle from the $\text{BIG}(\mathcal{T})$ values, we define $c(\tau(X))$ for Exists-CC-MMS as $\max\{0, 9E^2 + 6E - \text{Size}(\tau(X))\}$. Next, $\sigma^-(\tau(X)) := \min\{c(\tau(X)), \sigma^-\}$. σ^- is the sum of the values of one chore-copy of every good. Thus, $\sigma^-(\tau(X))$ is the maximum surplus in X for adding SMALL chores, whose values are in the set $\text{SMALL}(\mathcal{V}^g)$.

With two constraints using these functions, the IP ensures that we can allocate all the SMALL items. Using the notation from the $\mu \geq 0$ case, the IP for Exists-CC-MMS is as follows.

$$\sum_{j=1}^{\Gamma} x_j = n; \quad x_j \in \{0\} \cup \mathbb{N}, \forall j \in [\Gamma] \quad (6.6)$$

$$\sum_{j=1}^{\Gamma} \underline{\tau}_r^{(j)} x_j \geq (n-1)n_r^+, \quad \forall r \in [3E, 9E^2] \quad (6.7)$$

$$\sum_{j=1}^{\Gamma} \bar{\tau}_r^{(j)} x_j \geq n_r^-, \quad \forall r \in [3E, 9E^2] \quad (6.8)$$

$$\sum_{j=1}^{\Gamma} c(\tau^{(j)}) x_j \geq \sigma; \quad \sum_{j=1}^{\Gamma} \sigma^-(\tau^{(j)}) x_j \geq \sigma^- \quad (6.9)$$

We prove that Exists-CC-MMS solves this IP and returns an allocation which gives every agent a bundle of value at most $9E^2 + 9E$, if $\Delta \geq \text{CC-MMS}$.

The detailed algorithm for CC-MMS and its analysis are described in the next section.

6.4.2 Detailed discussion

Since we consider the negated value of chores, i.e., their cost, all chores have a non-negative cost. All associated values MMS, CC-MMS are also non-negative. Two small but key implications of this are (a) because of this negation, the MMS value is *at least* the average sum of all values (b) the sum of values of all items of the B-MMS instance is $v^- - v^+$.

We now introduce all notation for the CC-MMS problem and its formal definition.

Definition 6.25 (CC-MMS instance). *A tuple $\langle N, \mathcal{C}, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle$, where N is a set of agents, M is a set of chores, $(\mathcal{S}_j)_{j \in [m^+]}$ are m^+ sets of chores, each containing $(n - 1)$ identical copies of a good, and $u : \mathcal{C} \cup (\mathcal{S}_j)_{j \in [m^+]} \rightarrow \mathbb{R}_+$ is the identical additive valuation function of the agents in N for all items $\mathcal{C} \cup (\mathcal{S}_j)_{j \in [m^+]}$.*

Definition 6.26 (Valid allocation). *Given a CC-MMS instance $\mathcal{I} = \langle n, \mathcal{C}, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle$, an allocation A^π is valid if no agent receives more than 1 item from any set $\mathcal{S}_j \in (\mathcal{S}_j)_{j \in [m^+]}$, i.e., for all $i \in N, j \in [m^+]$, $|A^\pi_i \cap \mathcal{S}_j| \leq 1$.*

Definition 6.27 (CC-MMS value). *Given a CC-MMS instance $\mathcal{I} = \langle n, \mathcal{C}, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle$, let $\mathcal{F}$ be the set of all valid allocations. The CC-MMS value of the instance, denoted by CC-MMS, is defined as follows.*

$$\text{CC-MMS} = \operatorname{argmin}_{\mathcal{A} \in \mathcal{F}} \max_{A \in \mathcal{A}} u(A).$$

Finding allocations that give every agent a bundle worth at most her CC-MMS value is NP-Hard, as the MMS problem in a chore manna is a special case of this problem where $(\mathcal{S}_j)_{j \in [m^+]} = \emptyset$. Hence, we define approximate CC-MMS allocations as follows.

Definition 6.28 $((1 - \epsilon')\text{CC-MMS allocation})$. *Given a CC-MMS instance $\mathcal{I} = \langle n, \mathcal{C}, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle$, a valid allocation A^π is called a $(1 - \epsilon')$ CC-MMS allocation if*

$$u(A) \leq \text{CC-MMS} / (1 - \epsilon') \quad \forall A \in A^\pi.$$

Definition 6.29 (CC-MMS problem). *Given a CC-MMS instance $\mathcal{I} = \langle n, \mathcal{C}, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle$ and $\epsilon' > 0$, return a valid allocation A^π such that $\max_{A \in A^\pi} u(A) \leq \text{CC-MMS} / (1 - \epsilon')$.*

In Section 6.4.3, we show how to reduce the B-MMS problem to a CC-MMS problem, and why a PTAS for CC-MMS implies a PTAS for B-MMS.

6.4.3 Reducing B-MMS to CC-MMS

Given an instance $\mathcal{I} = \langle n, M, v \rangle$ we define the corresponding CC-MMS instance $\mathcal{I} = \langle n, \mathcal{C}, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle$ as follows. First, $\mathcal{C} = M^-$. Moreover, for all $j \in M^+$, define $\mathcal{S}_j$ to be a set of $(n - 1)$ chores, each represented by tuples as $\mathcal{S}_j := \{(j, k) \mid k \in [n - 1]\}$. Let $(\mathcal{S}_j)_{j \in [m^+]} := \bigcup_{j \in [m^+]} \mathcal{S}_j$ where $m^+ = |M^+|$. Finally, define $u(j) = |v(j)|$ for all $j \in M^-$ and $u((j, k)) = v(j)$ for all $j \in M^+$ and $k \in [n - 1]$. Let the chores in $\mathcal{S}_j$ be called chore-copies of good j .

Lemma 6.30. *Allocations of B-MMS are in one-to-one correspondence with valid allocations of CC-MMS, such that if allocation B^π of B-MMS corresponds to allocation C^π of CC-MMS, then $u(C_i) = v(B_i) + v^+, \forall i \in N$.*

Proof. Given a B-MMS allocation B^π , add chore-copies of each good to agents who did not receive the good in B^π , and discard all goods. This gives a valid CC-MMS allocation. The reverse allocation is obtained by similarly discarding all chore-copies and assigning the corresponding good to the agent who did not receive any chore-copy.

Every agent $i \in N$ receives in C^π all the chores assigned to her in B^π . Every good that was assigned to her in B^π is discarded in C^π . Due to this, the cost of her bundle (negated value) increases by the value of goods allotted to her in B^π . Further, for every good not assigned to her, she receives a chore-copy of it in C^π . Each chore-copy increases her cost by the value of the corresponding good. Hence, her cost increases by the value of all goods not assigned to her as well. Her bundle's total negated value in C^π is exactly, $\sum_{j \in M^+ \cap B_i} v(j) + \sum_{j \in M^+ \setminus B_i} v(j) = v^+$ more than in B^π . $\square$

Corollary 6.31. CC-MMS, relates to the MMS value of the B-MMS problem as,

$$\text{CC-MMS} = \mu + v^+. \quad (6.10)$$

Lemma 6.30 implies the following lemma.

Lemma 6.32. If $\mu \geq v^+/\rho$, a $(1-\epsilon')$ CC-MMS allocation implies a α -MMS allocation, for $\epsilon = \epsilon'(1+\rho)/(1+\rho\epsilon')$. Thus, an algorithm for the CC-MMS problem implies one for the B-MMS problem.

Proof. We take the $(1-\epsilon')$ CC-MMS allocation, say C^π , and consider the B-MMS allocation B^π corresponding to it, according to the one-to-one correspondence described in the proof of Lemma 6.30. From Equation (6.10), the largest bundle in B^π has value at most $\text{CC-MMS}/(1-\epsilon') - v^+$.

If $\mu \geq v^+/\rho$, we have, $\text{CC-MMS}/(1-\epsilon') - v^+ = (\mu + v^+)/(1-\epsilon') - v^+ = \mu/(1-\epsilon') + v^+(\epsilon'/(1-\epsilon')) \leq \mu/(1-\epsilon') + (\epsilon'/(1-\epsilon'))\rho\mu = (1+\rho\epsilon')(1-\epsilon')\mu$. To have this value at most $\mu/(1-\epsilon)$, we solve $(1+\rho\epsilon')(1-\epsilon') = 1/(1-\epsilon)$, and get $\epsilon = \epsilon'(1+\rho)/(1+\rho\epsilon')$. As ρ is a constant in the B-MMS problem, ϵ also is.

Given an algorithm for the CC-MMS problem, we find a $(1-\epsilon')$ CC-MMS allocation for $\epsilon' = \epsilon/(1+(1-\epsilon)\rho)$. From the above relation, this is a α -MMS allocation solving B-MMS. $\square$

Lemma 6.32 shows that solving CC-MMS is sufficient to solve B-MMS when $v^- > v^+$ (the other case is explained in Section 6.3). Algorithm 19 outlines the PTAS for CC-MMS. As described in Section 6.4.1, the algorithm for CC-MMS will perform a search over a range of possible values of CC-MMS. We now define this range. First, we know $\mu \geq v^+/\rho$. We also have the following two trivial lower bounds on μ . First, $\mu \geq (v^- - v^+)/n$. Also, the MMS value is at least the value of the chore with the largest absolute value. This is because in any allocation, the largest chore, of value say v , has to be added to some bundle. This bundle has value at least v . Hence the value of the highest valued bundle, that is the MMS value, is at least v . Thus, $\mu \geq \max_{j \in M^-} v(j)$. Hence, the lower bound on the MMS value is $\max\{v^+/\rho, (v^- - v^+)/n, \max_{j \in M^-} v(j)\}$. A trivial upper bound on the MMS value is the sum of all items, $(v^- - v^+)$. From Lemma 6.30, the corresponding bounds for CC-MMS are $\text{CC-MMS} \geq v^+ + \max\{v^+/\rho, (v^- - v^+)/n, \max_{j \in M^-} v(j)\}$, and $\text{CC-MMS} \leq (v^- - v^+) + v^+$.

Note that we want an allocation with the smallest cost of the largest bundle. Hence, the best value for MMS is the lower bound.

Each iteration of the search in Algorithm 19 takes a candidate value Δ in above range, and works as follows. Note that as we consider values greater than $\max_j v_j$ in the search, every item has value at most Δ in every iteration. The algorithm calls a subroutine **Exists-GC-MMS**, which checks if there is a valid allocation where all bundles have value at most $\Delta/(1-\bar{\epsilon})$ for $\bar{\epsilon} = \epsilon'/2$. If such an allocation exists the algorithm returns

it and if it does not exist it correctly reports $\Delta < \text{CC-MMS}$. We will discuss the **Exists-CC-MMS** algorithm in Section 6.4.4 and prove the following theorem.

Theorem 6.33. **Exists-CC-MMS**($\mathcal{I} = \langle n, C, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle, \epsilon, \Delta$) returns a tuple (A^π, flag) , where A^π is a $(1 - \epsilon')$ CC-MMS allocation and $\text{flag} = \text{true}$ when $\Delta \geq \text{CC-MMS}$, and runs in $O(mn)$ time.

Algorithm 19: Algorithm for CC-MMS

Input : $\mathcal{I} = \langle n, C, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle, \epsilon' > 0$
Output : $(1 - \epsilon')$ CC-MMS allocation if $\text{CC-MMS} \geq (1 + 1/\rho)v^+$

```

1  $\bar{\epsilon} \leftarrow \epsilon'/2, \Delta \leftarrow \max\{v^+/\rho, \max_j v(j), (v^- - v^+)/n\}$ 
2 while  $\Delta \leq v^-$  do
3    $(A^\pi, \text{flag}) \leftarrow \text{Exists-CC-MMS}(\mathcal{I} = \langle n, C, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle, \bar{\epsilon}, \Delta)$ 
4   if  $\text{flag}$  then
5     return  $A^\pi$ 
6   else
7      $\Delta \leftarrow \Delta/(1 - \bar{\epsilon})$ 
8  $A^\pi = (A_1, \dots, A_n)$  where  $A_i = \{(j, i) : \forall j \in [m^+]\}$  for  $i \in [n - 1], A_n = C$  // agents 1 to  $n - 1$ 
   each get one chore-copy of all goods.
9 return  $A^\pi$ 

```

Exists-CC-MMS, together with Algorithm 19, is a PTAS for CC-MMS. We search in the range defined above and return a trivial allocation if the value is not found.

Finally, Theorem 6.33 and Lemma 6.32 are used to show that Algorithm 19 gives an efficient algorithm for the B-MMS problem.

Lemma 6.34. For any CC-MMS instance, Algorithm 19 returns $(1 - \epsilon')$ CC-MMS allocation.

Proof. While running Algorithm 19, let Δ^* be the Δ for which **Exists-CC-MMS** returns $\text{flag} = \text{true}$. From Theorem 6.33, the corresponding allocation A^π returned by **Exists-CC-MMS** has all bundles of value at most $\Delta^*/(1 - \bar{\epsilon})$. If **Exists-CC-MMS** returns $\text{flag} = \text{true}$ for the first value of Δ considered, then as the first value is the lower bound on Δ , we have $\text{CC-MMS} \geq \Delta^*$, hence $\Delta^*/(1 - \bar{\epsilon}) \leq \text{CC-MMS}/(1 - \bar{\epsilon}) \leq \text{CC-MMS}/(1 - \epsilon')$, where the last inequality follows as $\bar{\epsilon} = \epsilon'/2$.

Otherwise, for the value of Δ considered in the previous iteration before Δ^* , that is, for $\Delta^*(1 - \bar{\epsilon})$, **Exists-CC-MMS** returned $\text{flag} = \text{false}$. Hence $\text{CC-MMS} > \Delta^*(1 - \bar{\epsilon})$. Hence, the largest valued bundle in A^π has value at least $\Delta^*/(1 - \bar{\epsilon}) \leq \text{CC-MMS}/(1 - \bar{\epsilon})^2 \leq \text{CC-MMS}/(1 - \epsilon')$.

Therefore, in both cases, A^π is a α -MMS-like allocation. $\square$

Lemmas 6.32 and 6.34, together with Theorem 6.33, prove the second half of Theorem 6.1, specified as follows.

Theorem 6.35. There is an algorithm for the B-MMS problem for the case $\mu < 0$ (before negating valuations) that runs in time $O(mnL)$, where L is the bit-length of the input.

It now remains to discuss the Algorithm **Exists-CC-MMS** and prove Theorem 6.33. The next section discusses this.

6.4.4 Algorithm for Exists-CC-MMS

In this section, we describe the algorithm **Exists-CC-MMS**, which given a **CC-MMS** instance, a value Δ , and a constant $\bar{\epsilon} > 0$, either outputs a valid allocation where all bundles have value at least $\Delta/(1 - \bar{\epsilon})$ or correctly reports $\Delta < \text{CC-MMS}$.

The Algorithm has two steps 1) Pre-processing and 2) Main Algorithm.

Pre-processing.(Algorithm 20, line 1) Let $E := \lfloor \frac{1}{\bar{\epsilon}} \rfloor$. Note that E is a constant integer that only depends on $\bar{\epsilon}$ and not on parameters in the **CC-MMS** instance. Scale the valuations v by $9E^2/\Delta$.

Let $\mathcal{V}^c$ and $\mathcal{V}^g$ be multi-sets of numbers corresponding to scaled valuations, respectively of all the chores in $\mathcal{C}$, and all the chore-copies in $(\mathcal{S}_j)_{j \in [m^+]}$. Formally, we define $\mathcal{V}^c, \mathcal{V}^g$ and sets G_j for $j \in [m^+]$ as follows.

$$\begin{aligned} c_j &:= u(j) \cdot \left(\frac{9E^2}{\Delta} \right), j \in \mathcal{C}, \\ g_j^k &:= u(j, k) \cdot \left(\frac{9E^2}{\Delta} \right), \forall (j, k) \in \mathcal{S}_j, \forall \mathcal{S}_j \in (\mathcal{S}_j)_{j \in [m^+]} \\ G_j &:= \{g_j^1, \dots, g_j^{n-1}\}, j \in M^+ \\ \mathcal{V}^c &:= \{c_1, \dots, c_{m-}\}, \quad \mathcal{V}^g := \bigcup_{j \in [m^+]} G_j. \end{aligned}$$

Finally, let $\mathcal{T} = \mathcal{V}^g \cup \mathcal{V}^c$.

This completes the pre-processing step. The following lemmas characterize partitions of $\mathcal{T}$ that correspond to approximately optimal **CC-MMS** allocations.

Definition 6.36 (Valid Partition of $\mathcal{T}$). *We call a partition $P = (P_1, \dots, P_n)$ of values in $\mathcal{T}$ valid if each P_k contains at most one element from each G_j , i.e., $|P_k \cap G_j| \leq 1$ for all $k \in [n]$ and $j \in [m^+]$.*

It is easy to see that each valid partition of $\mathcal{T}$ is equivalent to a valid allocation in its corresponding **CC-MMS** instance. The next lemma is a direct implication of this observation.

Lemma 6.37. *Given a **CC-MMS** instance $\mathcal{I} = \langle n, \mathcal{C}, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle$, if $\Delta \geq \text{CC-MMS}$ then there is a valid partition of $\mathcal{T}$ where the sum of values in each part is at most $9E^2$.*

Proof. Since $\Delta \geq \text{CC-MMS}$, there is a valid allocation for $\mathcal{I} = \langle n, \mathcal{C}, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle$ where all bundles have value at most Δ . If we scale the value of each item by $9E^2/\Delta$, the value of all bundles will also be scaled by $9E^2/\Delta$ because v is additive. Hence we get a valid partition where all parts have a total sum of at most $9E^2$. $\square$

Lemma 6.38. *Finding a valid partition of $\mathcal{T}$ where the sum of values in each part is at most $9(E^2 + E)$ is equivalent to finding a valid allocation for its corresponding **CC-MMS** instance where each bundle has value at most $\Delta/(1 - \bar{\epsilon})$.*

Proof. Let $P = (P_1, \dots, P_n)$ be the valid partition and let $A = (A_1, \dots, A_n)$ be its corresponding valid

allocation. We have $\sum_{p \in P_i} p = (9E^2/\Delta)u(A_i)$. We have

$$\begin{aligned}
& \sum_{p \in P_i} p \leq 9(E^2 + E) & \forall i \in [n] \\
& \left(\frac{9E^2}{\Delta}\right)u(A_i) \leq 9(E^2 + E) & \forall i \in [n] \\
\Rightarrow & u(A_i) \leq \Delta \left(\frac{9E^2 + 9E}{9E^2}\right) & \forall i \in [n] \\
\Rightarrow & u(A_i) \leq \Delta \left(1 + \frac{1}{E}\right) & \forall i \in [n] \\
\Rightarrow & u(A_i) \leq (1 + \bar{\epsilon})\Delta & \forall i \in [n]. \\
\Rightarrow & u(A_i) \leq \Delta/(1 - \bar{\epsilon}) & \forall i \in [n].
\end{aligned}$$

The second-last inequality holds because $E = \lfloor \frac{1}{\bar{\epsilon}} \rfloor \leq \frac{1}{\bar{\epsilon}}$. $\square$

Main Algorithm. This step obtains a valid partition of $\mathcal{T}$ where the sum of values in each part is at most $9(E^2 + E)$, if $\Delta \geq \text{CC-MMS}$, else correctly reports this by returning $flag = false$. Note that after scaling by $9E^2$, we can assume $t \leq 9E^2 \forall t \in \mathcal{T}$.

The key of the algorithm is an IP to allocate the **BIG** items. We first explain the IP.

Notation. We define **SMALL** and **BIG** values in $\mathcal{T}$. Call a value $t \in \mathcal{T}$ **SMALL** if $t < 3E$ and **BIG** if $t \geq 3E$. For each $T \subseteq \mathcal{T}$ let $\text{SMALL}(T)$ be the set of all small values in T and $\text{BIG}(T)$ be the set of all big values in T . We abuse notation and call a set $G_j \in \mathcal{V}^g$ *small* if it contains **SMALL** values and *big* otherwise². Let σ be the sum of all values t in $\text{SMALL}(\mathcal{T})$, i.e., $\sigma := \sum_{t \in \text{SMALL}(\mathcal{T})} t$. Let $\sigma^- := \sum_{t \in \text{SMALL}(\mathcal{V}^c)} t$ be the total sum of all values t in $\text{SMALL}(\mathcal{V}^c)$, and $\sigma^+ := (\sum_{t \in \text{SMALL}(\mathcal{V}^g)} t)/(n-1)$ be $1/(n-1)$ of the sum of **SMALL** values in $\mathcal{V}^g$. Note that σ^+ is equal to the sum of values obtained by picking one value from each small G_j , and $\sigma = \sigma^- + (n-1)\sigma^+$.

Next, we know that every **BIG** value will be in the range $[3E, 9E^2]$. Consider all integers r in $[3E, 9E^2]$. Let n_r^- be the number of values in $\text{BIG}(\mathcal{V}^c)$ with $\lceil g_j \rceil = r$ and let n_r^+ be the number of *big* G_j sets which have $\lceil g_j^k \rceil = r$ for all $k \in [n-1]$. Thus, there are $(n-1)n_r^+ + n_r^-$ items in $\mathcal{V}^g \cup \mathcal{V}^c$ with values whose integral part is r .

We now define the notation to represent a subset of **BIG** values and their sum. Let X denote a part in a partition of $\mathcal{T}$. We define the *type* of X by $\tau(X) = \langle \underline{\tau}(X), \bar{\tau}(X) \rangle = (\underline{\tau}_{3E}, \underline{\tau}_{3E+1}, \dots, \underline{\tau}_{9E^2}, \bar{\tau}_{3E}, \bar{\tau}_{3E+1}, \dots, \bar{\tau}_{9E^2})$; here $\underline{\tau}_r$ is the number of values in $\text{BIG}(X \cap \mathcal{V}^c)$ with integer part r and $\bar{\tau}_r$ is the number of values in $\text{BIG}(X \cap \mathcal{V}^g)$ with integer part r . Let $\text{SIZE}(\tau(X)) := \sum_{r=3E}^{9E^2} r(\underline{\tau}_r + \bar{\tau}_r)$ be the total sum of these rounded values in $\text{BIG}(\mathcal{T} \cap X)$.

Using this notation, we design an IP to find an assignment of **BIG** values in a $(1 - \bar{\epsilon})\text{CC-MMS}$ partition. That is, a selection of n types that allocate all **BIG** values such that, there is a valid way to add **SMALL** values and obtain an allocation where each part has value at most $9(E^2 + E)$. Thus, we will only consider types whose size $\text{SIZE}(\cdot)$ is at most $9E^2 + 9E$.

The variables of the IP correspond to all types that satisfy (i) $\text{SIZE}(\tau^{(j)}) \leq 9E^2 + 9E$, (ii) $\underline{\tau}_r^{(j)} \leq n_r^-$, (iii) $\bar{\tau}_r^{(j)} \leq n_r^+$. That is, every type represents a subset of **BIG** values that have at most n_r^- values from $\mathcal{V}^g$ and n_r^+ values from $\mathcal{V}^c$ with integral part of value r . Let us call the items of **CC-MMS** corresponding to the **BIG** (resp. **SMALL**) values as **BIG** (resp. **SMALL**) items. Intuitively, we allow types that represent valid allocations of

²Note that each G_j , $j \in [m^+]$ contains $(n-1)$ equal values. i.e., for each G_j , either $\text{SMALL}(G_j) = \emptyset$ or $\text{SMALL}(G_j) = G_j$.

BIG items of the CC-MMS instance. In the solution, every IP variable takes an integer value equal to the number of times the corresponding type is selected. This in turn represents the number of parts in the output allocation that have a subset of BIG items as represented by this type.

Lemma 6.39. *The IP has $O(1)$ number of variables.*

Proof. Every type with size at most $9E^2 + 9E$ can have at most $3E + 3$ BIG values, as every BIG value is at least $3E$. Each value is one of $[3E, 9E^2]$, a constant sized set. Hence, the number of types are at most $(9E^2 - 3E + 1)^{(3E+3)}$, which is a constant as E is a constant. The number of variables of the IP is at most the number of types with size at most $9E^2 + 9E$, hence is constant. $\square$

Before defining the IP, we define two *cost* functions for every type. The intuition for this function is described in the technical overview (Section 6.4.1). These are used to define constraints to allocate SMALL items.

Define $c(\tau(X)) := \{0, 9E^2 + 6E - \text{SIZE}(\tau(X))\}$, and $\sigma^-(\tau(X)) := \min\{c(\tau(X)), \sigma^-\}$.

Using the above notation, the IP is as defined in Section 6.4.1.

Algorithm 20: Exists-CC-MMS

Input : $\mathcal{I} = \langle n, C, (\mathcal{S}_j)_{j \in [m^+]}, u \rangle, \bar{\epsilon}, \Delta$
Output : (A^π, True) if there exists a $(1 - \bar{\epsilon})$ -CC-MMS allocation A^π and $(\emptyset, \text{False})$ otherwise

- 1 $\mathcal{V}^g \leftarrow \bigcup_{j \in [m^-]} G_j, G_j := \{g_j^1, \dots, g_j^{n-1}\}; g_j^k = u(j, k), \forall (j, k) \in \mathcal{S}_j, \forall \mathcal{S}_j \in (\mathcal{S}_j)_{j \in [m^-]}$
 $\mathcal{V}^c \leftarrow \{c_1, \dots, c_{m^+}\}, c_j = u(j). \left(\frac{9E^2}{\Delta}\right), j \in G; \mathcal{T} \leftarrow \mathcal{V}^g \cup \mathcal{V}^c$
- 2 $P = \{P_1, \dots, P_n\} \leftarrow (\emptyset, \dots, \emptyset)$
- 3 **if** IP has a solution X for $\mathcal{T}$ **then**
- 4 $j \leftarrow 1$
- 5 **for all** $i : x_i \neq 0$: **do**
- 6 Add BIG values to each $P_k, k \in [j, j + x_i]$ as per $\tau^{(i)}$; $j \leftarrow j + x_i + 1$
- 7 **while** there is $i : \sigma^-(\tau^i) > 0$ and some unallocated value from $\mathcal{V}^g$ **do**
- 8 **while** $\exists j : G_j \cap P_i \neq \emptyset$ and $\sigma^-(\tau^i) > 0$ **do**
- 9 assign one value from G_j to P_i
- 10 $\sigma^-(\tau^i) \leftarrow \sigma^-(\tau^i) - u(j)$
- 11 **while** $\exists j : G_j \cap P_i = \emptyset$ **do**
- 12 assign one value from G_j to each part in $N \setminus \{i\}$
- 13 **while** there is an unallocated $j \in \mathcal{V}^c$ **do**
- 14 for any $i : c(\tau^i) > 0 : P_i \leftarrow P_i \cup \{j\}$
- 15 $c(\tau^i) \leftarrow \max\{c(\tau^i) - u(j), 0\}$
- 16 $A \leftarrow$ Allocation of items corresponding to P **return** (A, True)
- 17 **return** $(\emptyset, \text{False})$

The Exists-CC-MMS algorithm is as follows. After applying the pre-processing step, it defines and solves the IP from Section 6.4.1. If the IP has a solution, then first it considers the items from the CC-MMS instance that correspond to the BIG values in $\mathcal{T}$. The algorithm partitions these items in n bundles by creating x_i bundles corresponding to type $\tau^{(i)}$. After this, it considers the bundles of BIG items that do not have total sum of values at most $9E^2 + 9E$. To each of these, it first adds the SMALL items corresponding to the small G_j subsets, by adding at most one item from each subset G_j , in any order. After all items from all sets G_j are allocated, it considers the parts whose sum is still less than $9E^2 + 9E$. It adds items corresponding to

SMALL($\mathcal{V}^c$) to any of these, until all items are exhausted. The algorithm returns the tuple $A^\pi, flag$, where A^π is the allocation formed by this process, and $flag = true$.

If the IP does not have a solution, it returns the tuple $(\emptyset, flag = false)$.

Algorithm 20 formally describes Exists-CC-MMS. We now analyze the correctness of Exists-CC-MMS, and the run time of Algorithm 19.

Lemma 6.40. *If $\Delta \geq \text{CC-MMS}$, then the IP has a solution.*

Proof. Lemma 6.37 shows that there is a partition of $\mathcal{T}$ where all bundles have value at most $9E^2$, as $\Delta \geq \text{CC-MMS}$. Let $T^* = (T_1^*, \dots, T_n^*)$ be this partition. Let $\tau^* = [\tau^1, \tau^2, \dots, \tau^n] = [\tau(T_i^*)]_{(i \in [n])}$ be the multi-set of types of each part in T^* .

For τ^* , constraints (6.6), (6.7), and (6.8) hold by definition of a valid partition. For any $\tau^i \in \tau^*$ we have,

$$\begin{aligned} \sum_{t \in \text{SMALL}(T_i^*)} t &\leq 9E^2 - \sum_{t \in \text{BIG}(T_i^*)} t \\ &\leq 9E^2 + 3E - \text{SIZE}(\tau^i) \\ &\leq c(\tau^i). \end{aligned}$$

The second inequality holds because the SIZE function rounds up all values, and there are at most $3E$ BIG values in each T_i^* . By adding the above inequality for all $\tau^i \in \tau^*$, we obtain (6.9) of the IP.

Since each T_i^* is a subset of a valid part, its corresponding type τ^i has at most one value from each C_j . Therefore, for any $\tau^i \in \tau^*$ we have,

$$\begin{aligned} \sum_{t \in \text{SMALL}(\tau^i \cap \mathcal{V}^g)} t &\leq c(\tau^i) - \sum_{t \in \text{SMALL}(\tau^i \cap \mathcal{V}^c)} t \\ &\leq c(\tau^i) - \sigma^- \leq \sigma^+(\tau^i). \end{aligned}$$

By adding the above inequality for all $T_i^* \in T^*$ we obtain the constraint (6.9) for IP. Thus, T^* is a solution of the IP. $\square$

Lemma 6.41. *If the IP has a solution, then the allocation returned by Exists-CC-MMS after finding this solution is a valid allocation of CC-MMS where all bundles have value at most $9E^2 + 9E$.*

Proof. Let τ^{sol} be the solution of the IP, P^{sol} be the partition of the BIG values corresponding to these types, and A^π be the allocation returned by Algorithm Exists-GC-MMS after finding τ^{sol} .

The algorithm adds SMALL items by first adding chore-copies. Initially, when all chore-copies are unallocated, as τ^{sol} satisfies constraint (b) of (6.9) of the IP, the total cost $\sigma^-(\tau(\cdot))$ for SMALL($\mathcal{V}^g$) for all bundles is at least σ^- . Hence, there is at least one agent with cost σ^- more than 0. The algorithm assigns as items from different sets $\mathcal{G}_j$ to this agent, until her cost $\sigma^-(\tau^i)$ becomes zero. At this point, either i has exhausted one chore-copy of every good, or her cost $c(\tau^i) = 0$. If the former is true, we have a smaller problem with $(n-2)$ copies of every good remaining, and n agents, hence by inductive reasoning, the algorithm assigns all chore-copies. If $c(\tau^i) = 0$, then first we assign one chore-copy of each good j that i did not get, to every remaining agent, to obtain a valid partition. Note that the sum of costs $\sigma^-(\tau^k)$ for the remaining agents $N \setminus \{i\}$, is equal to the space in each bundle for adding at most one copy of every chore, hence at least j , is more than the total cost of the unallocated items, including j , in $\mathcal{V}^g$, hence we can do this. The cost $\sigma^- \tau$ for all bundles now

is $\sum_{k \in N, k \neq i} \sigma^- \tau^k = \sum_{k \in N} \sigma^- \tau^k - \sigma^- \tau^i - (n-1) \sum_{j: G_j \cap A_i = \emptyset} u(j) \geq \sigma^- - \sigma^- \tau^i - (n-1) \sum_{j: G_j \cap A_i = \emptyset} u(j)$, which is exactly the cost of unallocated chore-copies. Hence, again by induction, we can repeat the process.

Finally, as constraint (6.9) is satisfied, until there is some unallocated chore, there will be some bundle A_i with $c(\tau(A_i)) > 0$. Line 8 and 14 can add one SMALL chore to A_i . Hence, all SMALL chores can be allocated. $\square$

Lemma 6.42. *Exists-CC-MMS runs in time $O(mn)$.*

Proof. The time to run Exists-CC-MMS is asymptotically equal to the time for constructing and solving the IP. Lenstra's algorithm [41] takes time exponential in the number of variables, $O(2^{1/\epsilon^2}) = O(2^{1/\epsilon})$ here, and polynomial in the largest coefficient of any variable in all inequalities, $m^- + (n-1)m^+ = O(mn)$ here. Hence, it requires $O(2^{1/\epsilon} mn) = O(mn)$ time to solve the IP. $\square$

6.5 Conclusions

In this chapter, we studied fair allocation of indivisible items in mixed manna under the notion of maximin share (MMS). In the mixed manna, finding even an approximate MMS value of an agent up to any approximation factor in $(0,1]$ is NP-Hard for general instances [38]. In this chapter, we complemented the hardness result by obtaining a PTAS to compute the MMS value, when its absolute value is at least $\frac{1}{\rho}$ times either the total value of all the goods or total cost of all the chores, for some constant ρ valued at least 1.

References

- [1] Nima AhmadiPourAnari et al. “Equilibrium pricing with positive externalities”. In: *Theoret. Comput. Sci.* 476 (2013), pp. 1–15.
- [2] Martin Aleksandrov. “Jealousy-freeness and other common properties in Fair Division of Mixed Manna”. In: *arXiv preprint arXiv:2004.11469* (2020).
- [3] Martin Aleksandrov and Toby Walsh. “Greedy Algorithms for Fair Division of Mixed Manna”. In: *CoRR* abs/1911.11005 (2019). arXiv: [1911.11005](https://arxiv.org/abs/1911.11005). URL: <http://arxiv.org/abs/1911.11005>.
- [4] Martin Aleksandrov and Toby Walsh. “Two Algorithms for Additive and Fair Division of Mixed Manna”. In: *German Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer. 2020, pp. 3–17.
- [5] Georgios Amanatidis, Georgios Birmpas, and Evangelos Markakis. “On Truthful Mechanisms for Maximin Share Allocations”. In: *Proc. 25th Intl. Joint Conf. Artif. Intell. (IJCAI)*. Ed. by Subbarao Kambhampati. 2016, pp. 31–37.
- [6] Georgios Amanatidis et al. “Approximation Algorithms for Computing Maximin Share Allocations”. In: *ACM Trans. Algorithms* 13.4 (2017), 52:1–52:28.
- [7] Georgios Amanatidis et al. “Truthful Allocation Mechanisms Without Payments: Characterization and Implications on Fairness”. In: *Proc. 18th Conf. Economics and Computation (EC)*. 2017, pp. 545–562.
- [8] Haris Aziz, Hau Chan, and Bo Li. “Maxmin Share Fair Allocation of Indivisible Chores to Asymmetric Agents”. In: *International Conference on Autonomous Agents and MultiAgent Systems, AAMAS ’19*. 2019, pp. 1787–1789. URL: <http://dl.acm.org/citation.cfm?id=3331919>.
- [9] Haris Aziz, Hau Chan, and Bo Li. “Weighted maxmin fair share allocation of indivisible chores”. In: *arXiv preprint arXiv:1906.07602* (2019).
- [10] Haris Aziz, Bo Li, and Xiaowei Wu. “Strategyproof and Approximately Maxmin Fair Share Allocation of Chores”. In: *Proc. 28th Intl. Joint Conf. Artif. Intell. (IJCAI)*. 2019, pp. 60–66.
- [11] Haris Aziz et al. “Algorithms for Max-Min Share Fair Allocation of Indivisible Chores”. In: *Proc. 31st Conf. Artif. Intell. (AAAI)*. 2017, pp. 335–341.
- [12] Siddharth Barman and Sanath Kumar Krishnamurthy. “Approximation algorithms for maximin fair division”. In: *Proc. 18th Conf. Economics and Computation (EC)*. 2017, pp. 647–664.
- [13] Siddharth Barman et al. “Fair Division of Indivisible Goods Among Strategic Agents”. In: *Proc. 18th Conf. Auton. Agents and Multi-Agent Systems (AAMAS)*. 2019, pp. 1811–1813.
- [14] Siddharth Barman et al. “Groupwise Maximin Fair Allocation of Indivisible Goods”. In: *Proc. 32nd Conf. Artif. Intell. (AAAI)*. 2018, pp. 917–924.

- [15] Xiaohui Bei et al. “Connected Fair Allocation of Indivisible Goods”. In: *CoRR* (2019). arXiv: [1908.05433](#).
- [16] Arpita Biswas and Siddharth Barman. “Fair Division Under Cardinality Constraints”. In: *Proc. 27th Intl. Joint Conf. Artif. Intell. (IJCAI)*. 2018, pp. 91–97.
- [17] Anna Bogomolnaia et al. “Competitive division of a mixed manna”. In: *CoRR* (2017). arXiv: [1702.00616](#). URL: <http://arxiv.org/abs/1702.00616>.
- [18] Anna Bogomolnaia et al. “Dividing bads under additive utilities”. In: *Soc. Choice Welf.* 52(3) (2019), pp. 395–417. DOI: [10.1007/s00355-018-1157-x](#). URL: <https://doi.org/10.1007/s00355-018-1157-x>.
- [19] Sylvain Bouveret and Michel Lemaître. “Characterizing conflicts in fair division of indivisible goods using a scale of criteria”. In: *Auton. Agents Multi Agent Syst.* 30.2 (2016), pp. 259–290.
- [20] Sylvain Bouveret and Michel Lemaître. “Efficiency and Sequenceability in Fair Division of Indivisible Goods with Additive Preferences”. In: *CoRR* (2016). arXiv: [1604.01734](#).
- [21] Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.
- [22] Simina Brânzei et al. “Matchings with externalities and attitudes”. In: *Proc. 12th Conf. Auton. Agents and Multi-Agent Systems (AAMAS)*. 2013, pp. 295–302.
- [23] Eric Budish. “The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes”. In: *J. Political Economy* 119.6 (2011), pp. 1061–1103.
- [24] Bhaskar Ray Chaudhury et al. “A little charity guarantees almost envy-freeness”. In: *Proc. 31st Symp. Discrete Algorithms (SODA)*. (accepted in SICOMP), 2020, pp. 2658–2672.
- [25] W Fernandez De La Vega and George S. Lueker. “Bin packing can be solved within $1 + \varepsilon$ in linear time”. In: *Combinatorica* 1.4 (1981), pp. 349–355.
- [26] Raul Etkin, Abhay Parekh, and David Tse. “Spectrum sharing for unlicensed bands”. In: *IEEE Journal on selected areas in communications* 25.3 (2007), pp. 517–528.
- [27] Alireza Farhadi et al. “Fair Allocation of Indivisible Goods to Asymmetric Agents”. In: *J. Artif. Intell. Res.* 64 (2019), pp. 1–20.
- [28] Jugal Garg, Peter McGlaughlin, and Setareh Taki. “Approximating Maximin Share Allocations”. In: *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019*. 2019, 20:1–20:11.
- [29] Jugal Garg and Setareh Taki. “An improved approximation algorithm for maximin shares”. In: *Artificial Intelligence* 300 (2021), p. 103547. ISSN: 0004-3702.
- [30] Vael Gates, Thomas L. Griffiths, and Anca D. Dragan. “How to Be Helpful to Multiple People at Once”. In: *Cognitive Science* 44 (2020).
- [31] Mohammad Ghodsi et al. “Fair Allocation of Indivisible Goods: Improvement”. In: *Math. Oper. Res.* 46.3 (2021), pp. 1038–1053.
- [32] Laurent Gourvès and Jérôme Monnot. “On maximin share allocations in matroids”. In: *Theoret. Comput. Sci.* 754 (2019), pp. 50–64.
- [33] Xin Huang and Pinyan Lu. “An Algorithmic Framework for Approximating Maximin Share Allocation of Chores”. In: *Proceedings of the 22nd ACM Conference on Economics and Computation*. EC ’21. Budapest, Hungary, 2021, pp. 630–631. ISBN: 9781450385541.

- [34] Klaus Jansen, Kim-Manuel Klein, and José Verschae. “Closing the Gap for Makespan Scheduling via Sparsification Techniques”. In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP*. Vol. 55. 2016, 72:1–72:13.
- [35] Klaus Jansen, Kim-Manuel Klein, and José Verschae. “Closing the Gap for Makespan Scheduling via Sparsification Techniques”. In: *Math. Oper. Res.* 45.4 (2020), pp. 1371–1392.
- [36] David S Johnson. “The NP-completeness column: An ongoing guide”. In: *Journal of Algorithms* 3.4 (1982), pp. 381–395.
- [37] Narendra Karmarkar and Richard M Karp. “An efficient approximation scheme for the one-dimensional bin-packing problem”. In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. IEEE. 1982, pp. 312–320.
- [38] Rucha Kulkarni, Ruta Mehta, and Setareh Taki. “Approximating Maximin Shares with Mixed Manna”. In: *CoRR* abs/2007.09133 (2020). arXiv: [2007.09133](https://arxiv.org/abs/2007.09133). URL: <https://arxiv.org/abs/2007.09133>.
- [39] David Kurokawa, Ariel D Procaccia, and Junxing Wang. “When can the maximin share guarantee be guaranteed?” In: *Proc. 30th Conf. Artif. Intell. (AAAI)*. 2016, pp. 523–529.
- [40] David Kurokawa, Ariel D. Procaccia, and Junxing Wang. “Fair Enough: Guaranteeing Approximate Maximin Shares”. In: *J. ACM* 65.2 (2018), 8:1–8:27.
- [41] Hendrik W Lenstra Jr. “Integer programming with a fixed number of variables”. In: *Mathematics of operations research* 8.4 (1983), pp. 538–548.
- [42] Zhentao Li and Adrian Vetta. “The Fair Division of Hereditary Set Systems”. In: *Proc. 14th Conf. Web and Internet Economics (WINE)*. 2018, pp. 297–311.
- [43] Zbigniew Lonc and Mirosław Truszczynski. “Maximin share allocations on cycles”. In: *CoRR* (2019). arXiv: [1905.03038](https://arxiv.org/abs/1905.03038).
- [44] Hervé Moulin. *Fair division and collective welfare*. MIT press, 2004.
- [45] Ariel D Procaccia and Junxing Wang. “Fair enough: Guaranteeing approximate maximin shares”. In: *Proc. 15th Conf. Economics and Computation (EC)*. 2014, pp. 675–692.
- [46] Hugo Steinhaus. “The problem of fair division”. In: *Econometrica* 16 (1948), pp. 101–104.
- [47] Thomas Vossen. “Fair allocation concepts in air traffic management”. PhD thesis. Supervisor: MO Ball, University of Maryland, College Park, Md, 2002.
- [48] Gerhard J Woeginger. “A polynomial-time approximation scheme for maximizing the minimum machine completion time”. In: *Oper. Res. Lett.* 20.4 (1997), pp. 149–154.