ABSTRACT

Title of dissertation: Fairness Guarantees in Allocation Problems

Hadi Yami
Doctor of Philosophy, 2020

Dissertation directed by: Professor MohammadTaghi Hajiaghayi
Computer Science Department

Fair division problems have been vastly studied in the past 60 years. This line of research was initiated by the work of Steinhaus in 1948 in which the author introduced the cake cutting problem as follows: given a heterogeneous cake and a set of agents with different valuation functions, the goal is to find a fair allocation of the cake to the agents. In order to study this problem, several notions of fairness are proposed, the most famous of which are proportionality and envy-freeness, introduced by Steinhaus in 1948 and Foley in 1967. The fair allocation problems have been studied in both divisible and indivisible settings.

For the divisible setting, we explore the ”Chore Division Problem”. The chore division problem is the problem of fairly dividing an object deemed undesirable among a number of agents. The object is possibly heterogeneous, and hence agents may have different valuations for different parts of the object. Chore division is the dual problem of the celebrated cake cutting problem. We give the first discrete and bounded envy-free chore division protocol for any number of agents.

For the indivisible setting, we use the maximin share paradigm introduced by Budish as a measure of fairness. We improve previous results on this measure of fairness in the additive setting and generalize our results for submodular, fractionally subadditive, as well as subadditive settings. We also model the maxmin share fairness paradigm for indivisible goods with different entitlements.

For the indivisible setting, we also consider the most studied notion of fairness, envy-freeness. It is known that envy-freeness cannot be always guaranteed in the allocation of indivisible items. We suggest envy-freeness up to a random item (EFR) property which is a relaxation of envy-freeness up to any item (EFX) and give an approximation guarantee. For this notion, we provide a polynomial-time 0.72-approximation allocation algorithm.

Fairness Guarantees in Allocation Problems

by

Hadi Yami

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Dedication

To my lovely parents

Acknowledgments

First and foremost, I would like to thank my adviser MohammadTaghi Hajiaghayi for all the time that we spent working together. He patiently allowed me to follow my interests, to freely fail time to time, and then helped me to learn from my failures. He has been always there to advise me even during the chaotic moments just before the deadlines. I always felt to have his full support and have learned many life lessons from him. He has always made himself available for help and advice and there has never been an occasion when I knocked on his door and he did not give me time. I learned from Mohammad to be a fighter in research. Mohammad and I have worked on many challenging problems together, and I rarely ever saw him give up on finding a solution. Neither has he ever lost faith in any of his students during the ups and downs of a PhD life.

Many thanks to Lawrence Ausubel, John Dickerson, William Gasarch, MohammadTaghi Hajiaghayi, David Mount, and Mihai Pop for devoting their time and serving on my thesis committee, and for their frequent advice and suggestions.

I had the privilege to work with many bright researchers throughout my Ph.D. I would like to express my strong gratitude to all my collaborators and co-authors who influenced my works with their diverse backgrounds and knowledge.

I would also greatly thank all my old and new friends for their support. I would like to thank All my great non-UMD friends Sepehr Abbasizadeh, Pooyan Ehsani, Farshid Effati, Mehrdad Khani, Elham Shakeri, and Ali Shameli. All my friends in Maryland: Saba Ahmadi, Madeline Alizadeh, Rouzbeh Basirian, Soheil Behnezhad, Sina Dehghani, Mahsa Derakhashan, Soheil Ehsani, Hossein Esfandiari, Alireza Farhadi, Sepehr Ghader, Amin Ghiasi, Milad Gholami, MohammadReza Khani, Sahar Khosravi, Mahyar Najibi, kiana Roshanzamir, Parsa Saadatpanah, Hamed Saleh, Saeed Seddighin, and Ali Shafahi.

Last but most importantly, I dedicate this dissertation to my family. To my lovely parents Roghayeh and Mousa, without whom this journey was not even possible. My brothers, Hamed and Yaser, my sisters in law, Farzaneh and Lida, who always backed me up unconditionally, and their lovely children Aisou and Ryan. I was greatly blessed to have them by my side through all these years and cannot find words adequate enough to truly thank them.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

One of the great challenges of our era is a thorough understanding of fairness. The problem of fairly allocating limited natural resources such as fossil fuels and clean water is always a great challenge in the world. Fair division is considered as one of the important active areas in microeconomics and algorithmic game theory today. The literature usually distinguishes between allocation of indivisible goods and divisible goods. The application of the fair allocation of divisible goods is when the goal is to fairly divide land, time, memory on a computer, or any other divisible item or chore. For instance, after the collapse of Germany in 1945, the problem of fair separation of Berlin between the Soviet, French, British, and American zones was an important challenge. The fair allocation of indivisible goods is also important when the goal is to fairly allocate some indivisible items such as furniture or indivisible tasks. For instance, after any divorce, the court will divide property between spouses in a way that it considers fair.

This thesis is divided into four parts. In the first part, Chapter 2, we consider the problem of fair allocation for indivisible goods. We use *the maxmin share* paradigm introduced by Budish [1] as a measure for fairness. Kurokawa, Procaccia, and Wang [2] were the first to investigate this fundamental problem in the additive setting. They show that a maxmin guarantee (1-MMS allocation) is not always possible even when the number of agents is limited to 3. While the existence of an approximation solution (e.g. a $1/2$-MMS allocation) is quite straightforward, improving the guarantee becomes subtler for larger constants. Kurokawa *et al.* [2] provide a proof for existence of a $2/3$-MMS allocation and leave the question open for better guarantees.

Our main contribution is an answer to the above question. We improve the result of Kurokawa *et al.* to a $3/4$ factor in the additive setting. The main idea for our $3/4$-MMS allocation method is clustering the agents. To this end, we introduce three notions and techniques, namely *reducibility*, *matching allocation*, and *cycle-envy-freeness*, and prove the approximation guarantee of our algorithm via non-trivial applications of these techniques. Our analysis involves coloring and double counting arguments that might be of independent interest.

One major shortcoming of the current studies on fair allocation is the additivity assumption on the valuations. We alleviate this by extending our results to the case of submodular, fractionally subadditive, and subadditive settings. More precisely, we give constant approximation guarantees for submodular and XOS agents, and a logarithmic approximation for the case of subadditive agents. Furthermore, we complement our results by providing close upper bounds for each class of valuation functions. Finally, we present algorithms to find such allocations for additive, submodular, and XOS settings in polynomial time.

This work was published at EC-2018 [3].

In the second part, Chapter 3, we consider the problem of fairly allocating a set of indivisible items among a set of players. Envy-free up to one good ($\mathsf{EF1}$) and envy-free up to any good (EFX) are two well-known extensions of envy-freeness for the case of indivisible items. It is shown that EF1 can always be guaranteed for agents with subadditive valuations [4]. In sharp contrast, it is unknown whether or not an EFX allocation always exists, even for three agents and additive valuations. In addition, the best approximation guarantee for EFX is $(\phi - 1) \simeq 0.61$ by Amanitidis et al. [5].

In order to find a middle ground to bridge this gap, in this paper we suggest another fairness criterion, namely *envy-freeness up to a random good* or EFR, which is weaker than EFX, yet stronger than EF1. For this notion, we provide a polynomial-time 0.72-approximation allocation algorithm. For our algorithm we introduce Nash Social Welfare Matching which makes a connection between Nash Social Welfare and envy freeness. We believe Nash Social Welfare Matching will find its applications in future work.

The third part, Chapter 4, considers the chore division problem, introduced by Gardner in 1970s [6]. Chore division is the problem of fairly dividing a chore among n different agents. In particular, in an envy-free chore division, we would like to divide a negatively valued heterogeneous object among a number of agents who have different valuations for different parts of the object, such that no agent envies another agent. It is the dual variant of the celebrated cake cutting problem, in which we would like to divide a desirable object among agents. There has been an extensive amount of study and effort to design bounded and envy-free protocols/algorithms for fair division of chores and goods, such that envy-free cake cutting became “one of the most important open problems in 20-th century mathematics” according to [7]. However, despite persistent efforts, due to delicate nature of the problem, there was no bounded protocol known for cake cutting even among four agents, until the breakthrough of [8], which provided the first discrete and bounded envy-free protocol for cake cutting for four agents. Afterward, [9], generalized their work and provided an envy-free cake cutting protocol for any number of agents to settle a significant and long-standing open problem. However, there is much less known for chore division. Unfortunately, there is no general method known to apply cake cutting techniques to chore division. Thus, it remained an open problem to find a discrete and bounded envy-free chore division protocol even for four agents.

We provide the first discrete and bounded envy-free protocol for chore division for an arbitrary number of agents. We produce major and powerful tools for designing protocols for the fair division of negatively valued objects. These tools are based on structural results and important observations. In general, we believe these structures and techniques may be useful not only in chore division but also in other fairness problems.

This work was published at SODA-2018 [10].

In the last part, Chapter 5, we study the fair allocation of indivisible goods to agents with *unequal entitlements*. Fair allocation has been the subject of many studies in both divisible and indivisible settings. Our emphasis is on the case where the goods are indivisible and agents have unequal entitlements. This problem is a generalization of the work by Kurokawa, Procaccia, and Wang [2] wherein the agents are assumed to be symmetric with respect to their entitlements. Although Kurokawa, Procaccia, and Wang show an almost fair (constant approximation) allocation exists in their setting, our main result is in sharp contrast to their observation. We show that, in some cases with n agents, no allocation can guarantee better than $1/n$ approximation of a fair allocation when the entitlements are not necessarily equal. Furthermore, we devise a simple algorithm that ensures a $1/n$

approximation guarantee. We also run some experiments on real-world data and show that, in practice, a fair allocation is likely to exist.

This work was published at Journal of Artificial Intelligence Research [11].

Chapter 2: Fair Allocation of Indivisible Goods: Guaranteeing Approximate Maximin Shares

2.1 Introduction

Suppose we have a set of m indivisible items, and wish to distribute them among n agents. Agents have valuations for each set of items that are not necessarily identical. How hard is it to divide the items between the agents to make sure everyone receives a fair share?

Fair division problems have been vastly studied in the past 60 years, (see, e.g. [12, 13, 14, 1, 15, 2, 16]). This line of research was initiated by the work of [16] in which the author introduced the *cake cutting* problem as follows: given a heterogeneous cake and a set of agents with different valuation functions, the goal is to find a fair allocation of the cake to the agents.

In order to study this problem, several notions of fairness have been proposed, the most famous of which are *proportionality* and *envy-freeness*, introduced by [16] and Foley [17]. A division is called proportional, if the total value of the allocated pieces to each agent is at least $1/n$ fraction of his total value for the entire cake, where n is the number of agents. In an envy-free division, no agent wishes to exchange his share with another agent, i.e., every agent's valuation for his share is at least as much as his valuation for the other agents' shares. Clearly, proportionality is implied by envy-freeness.

Dubins and Spanier [15] propose a simple *moving knife* procedure that can guarantee a proportional division of the cake. For envy-freeness, Selfridge and Conway design an algorithm that guarantees envy-freeness when the number of agents is limited to 3. Later, Brams and Taylor extend this guarantee to an arbitrary number of agents in the additive setting [18]. However, their method for allocating the cake makes an un-bounded number of cuts. Recently, Aziz and Mackenzie proposed a bounded protocol for envy-free allocation of the cake [9].

The problem becomes even more subtle when we assume the items are indivisible. It is not hard to show that for indivisible items, neither proportionality nor envy-freeness can be guaranteed; for instance, when the number of items is smaller than the number of agents, at least one agent receives no items.

From a theoretical standpoint, proportionality and envy-freeness are too strong to be delivered in the case of indivisible goods. Therefore, Budish [1] proposed a newer notion of fairness for indivisible goods, namely *the maxmin share*, which has attracted a lot of attention in recent years [2, 12, 14, 19, 20, 11, 21, 22]. Imagine that we ask an agent a_i to partition a set $\mathcal{M}$ of m items into n bundles and collect the bundle with the smallest value. To maximize his profits, agent a_i tries to divide $\mathcal{M}$ in a way that maximizes the value of the bundle with the lowest value to him. Based on this, the maxmin share of an agent a_i, denoted by MMS_i, is the value of the least valuable bundle in agent a_i's allocation; that is, the maximum profit a_i can obtain in this procedure. Clearly, MMS_i is the most that can be guaranteed to an agent, since if all valuations are the same, at least one agent obtains a valuation of at most MMS_i from his allocated set. The question is then, whether there exists an allocation which guarantees MMS_i for every agent a_i? Therefore, we call an allocation MMS, if every agent a_i receives a collection of items that are together worth at least MMS_i to him. Bouveret and Lemaitre [14] showed that for the restricted cases, when the valuations of the items for each agent are either 0 or 1, or when $m \leq n + 3$, an MMS allocation is guaranteed to exist. In other words, each a_i can be guaranteed to receive a profit of at least MMS_i from his allocated items.

While the experiments support the existence of an MMS allocation in general [14], this conjecture was refuted by the pioneering work of Kurokawa, Procaccia, and Wang [2]. Kurokawa *et al.* [2] provided a surprising counter-example that admits no MMS allocation. They also show that a $2/3$-MMS allocation always exists, i.e. there exists an algorithm that allocates the items to the agents in such a way that every agent a_i receives a share that is worth at least $2/3\mathsf{MMS}_i$ to him. In particular, they show for $n \leq 4$, their algorithm finds a $3/4$-MMS allocation. However, their algorithm does not run in polynomial time unless we assume the number of agents is bounded by a constant number. Following this work, Amanatidis, Markakis, Nikzad, and Saberi [12], improve this result by presenting a polynomial time algorithm for finding a $(2/3 - \epsilon)$-MMS allocation to any number of agents for constant ϵ. However, the heart of their algorithm is the same as [2]. In addition to this, Amanatidis *et al.* prove that for $n = 3$, a $7/8$-MMS allocation is always possible. Note that, the counter example provided by Kurokawa *et al.*[2] requires a number of goods that is exponential to the number of agents. Kurokawa *et al.*in [23] provided a better construction for the counter-example with a linear number of goods.

In this work, we improve the result of Kurokawa *et al.*[2] by proving that a $3/4$-

MMS allocation always exists. We also give a polynomial time algorithm to find such an allocation. Of course, this only holds if the valuation of the agents for the items are additive. We further go beyond the additive setting and extend this result to the case of submodular, XOS, and subadditive settings. More precisely, we give constant approximation algorithms for submodular and XOS settings that run in polynomial time. For the subadditive case, we prove that a $1/10\lceil\log m\rceil$-MMS allocation is guaranteed to exist. We emphasize that finding the exact value of MMS_i for an agent is NP-hard. Furthermore, to the best of our knowledge, no PTAS is known for computing the MMS values in non-additive settings. Thus, any α-MMS allocation algorithm in non-additive settings must overcome the difficulty that the value of MMS_i is not known in advance. Therefore, our algorithms don't immediately follow from our existential proofs.

In order to present the results and techniques, we briefly state the fair allocation problem. Note that you can find a formal definition of the problem with more details in Section 3.2. The input to a maxmin fair allocation problem is a set $\mathcal{M}$ of m items and a set $\mathcal{N}$ of n agents. Fix an agent $a_i \in \mathcal{N}$ and let $V_i : 2^{\mathcal{M}} \to \mathbb{R}^+$ be the valuation function of a_i. Consider the set Π_r of all partitions of the items in $\mathcal{M}$ into r non-empty sets. We define $\mathsf{MMS}^r_{V_i}(\mathcal{M})$ as follows:

$$\mathsf{MMS}^r_{V_i}(\mathcal{M}) = \max_{P^*=\langle P^*_1,P^*_2,\ldots,P^*_r\rangle\in\Pi_r} \min_{1\le j\le r} V_i(P^*_j).$$

In the context of fair allocation, we denote the maxmin value of an agent a_i by $\mathsf{MMS}_i = \mathsf{MMS}^n_{V_i}(\mathcal{M})$. The fair allocation problem is defined as follows: *for a given parameter α, can we distribute the items among the agents in such a way that every agent a_i receives a* *set of items with a value of at least* $\alpha\mathsf{MMS}_i$ *to him?* Such an allocation is called an α-MMS allocation. We consider the fair allocation problem in both additive and non-additive settings (including submodular, XOS, and subadditive valuations). For non-additive settings, we use oracle queries to access the valuations. Note that, for non-additive settings, eliciting the entire valuation function of each agent needs an exponential number of queries. However, our methods for allocating the items in non-additive settings only use a polynomial number of queries.

There are many applications for finding fair allocations in the additive and non-additive settings. For example, *spliddit*, a popular fair division website[1] suggests indisputable and provably fair solutions for many real-world problems such as sharing rents, distributing tasks, dividing goods, etc. For dividing goods, *spliddit* uses the maximum Nash welfare allocation (the allocation that maximizes the product of utilities). However, the current best approximation guarantee and the state-of-the-art method for allocating indivisible goods is based on the result of [2] that guarantees a $2/3$-MMS allocation. We believe our results can improve their performance.

2.2 Preliminaries

Throughout this chapter we assume the set of agents is denoted by $\mathcal{N}$ and the set of items is referred to by $\mathcal{M}$. Let $|\mathcal{N}| = n$ and $|\mathcal{M}| = m$, we refer to the agents by a_i and to the items by b_i, i.e., $\mathcal{N} = \{a_1, a_2, \ldots, a_n\}$ and $\mathcal{M} = \{b_1, b_2, \ldots, b_m\}$. We denote the valuation of agent a_i for a set S of items by $V_i(S)$. Our interest is in valuation functions that are monotone and non-negative. More precisely, we assume $V_i(S) \geq 0$ for every agent a_i and

[1]`http://www.spliddit.org`

$S \subseteq \mathcal{M}$, and for every two sets S_1 and S_2 we have $V_i(S_1 \cup S_2) \geq \max\{V_i(S_1), V_i(S_2)\}$. Due to obvious impossibility results for the general valuation functions, we restrict our attention to four classes of set functions:

- **Additive**: A set function $V(.)$ is additive if $V(S_1)+V(S_2) = V(S_1 \cup S_2)+V(S_1 \cap S_2)$ for every two sets $S_1, S_2 \in \mathsf{ground}(V)$.

- **Submodular**: A set function $V(.)$ is submodular if $V(S_1)+V(S_2) \geq V(S_1 \cup S_2)+V(S_1 \cap S_2)$ for every two sets $S_1, S_2 \in \mathsf{ground}(V)$.

- **Fractionally Subadditive (XOS)**: An XOS set function $V(.)$ can be shown via a finite set of additive functions $\{V_1, V_2, \dots, V_\alpha\}$ where $V(S) = \max_{i=1}^{\alpha} V_i(S)$ for any set $S \subseteq \mathsf{ground}(V)$.

- **Subadditive**: A set function $V(.)$ is subadditive if $V(S_1) + V(S_2) \geq V(S_1 \cup S_2)$ for every two sets $S_1, S_2 \subseteq \mathsf{ground}(V)$.

For additive functions, we assume the value of the function for every element is given in the input. However, representing other classes of set functions requires access to oracles. For submodular functions, we assume we have access to *query oracle* defined below. Query oracles are great identifier for submodular functions, however, they are too weak when it comes to XOS and subadditive settings. For such functions, we use a stronger oracle which is called *demand oracle*. It is shown that for some functions, such as gross substitutes, a demand oracle can be implemented via a query oracle in polynomial time [24]. In addition to this, we consider a special oracle for XOS functions which is called *XOS oracle*. Access to query oracles for submodular functions, XOS oracle for

XOS functions, and demand oracles for XOS and subadditive functions are quite common and have been very fruitful in the literature [25, 26, 27, 28, 29, 24, 30]. In what follows, we formally define the oracles:

- **Query oracle:** Given a function f, a query oracle $\mathcal{O}$ is an algorithm that receives a set S as input and computes $f(S)$ in time $O(1)$.

- **Demand oracle:** Given a function f, a demand oracle $\mathcal{O}$ is an algorithm that receives a sequence of prices $p_1, p_2, \ldots, p_n$ as input and finds a set S such that $f(S) - \sum_{e \in S} p_e$ is maximized. We assume the running time of the algorithm is $O(1)$.

- **XOS oracle:** (defined only for an XOS functions f) Given a set S of items, it returns the additive representation of the function that is maximized for S. In other words, it reveals the contribution of each item in S to the value of $f(S)$.

Let Π_r be the set of all partitions of $\mathcal{M}$ into r disjoint subsets. For every r-partitioning $P^* \in \Pi_r$, we denote the partitions by $P^*_1, P^*_2, \ldots, P^*_r$. For a set function $f(.)$, we define $\mathsf{MMS}^r_f(\mathcal{M})$ as follows:

$$\mathsf{MMS}^r_f(\mathcal{M}) = \max_{P^* \in \Pi_r} \min_{1 \leq j \leq r} f(P^*_j).$$

For brevity we refer to $\mathsf{MMS}^n_{f_i}(\mathcal{M})$ by MMS_i.

An allocation of items to the agents is a vector $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$ where $\bigcup A_i = \mathcal{M}$ and $A_i \cap A_j = \varnothing$ for every two agents $a_i, a_j \in \mathcal{N}$. An allocation $\mathcal{A}$ is α-MMS, if every agent a_i receives a subset of the items whose value to that agent is at least α times

MMS_i. More precisely, $\mathcal{A}$ is α-MMS if and only if $V_i(A_i) \geq \alpha\mathsf{MMS}_i$ for every agent $a_i \in \mathcal{N}$.

We define the notion of *reducibility* for an instance of the problem as follows.

Definition 2.2.1 *We say an instance of the problem is α-reducible, if there exist a set $T \subset \mathcal{N}$ of agents, a set S of items, and an allocation $\mathcal{A} = \langle A_1, A_2, \ldots, A_{|T|}\rangle$ of S to agents of T such that*

$$\forall a_i \in T \qquad\qquad V_i(A_i) \geq \alpha\mathsf{MMS}_i$$

and

$$\forall a_i \notin T \qquad \mathsf{MMS}_{V_i}^{n-|T|}(\mathcal{M} \setminus S) \geq \mathsf{MMS}_i.$$

Similarly, we call an instance *α-irreducible* if it is not α-reducible. The intuition behind Definition 2.2.1 is the following: In order to prove the existence of an α-MMS allocation for every instance of the problem, it only suffices to prove this for the α-irreducible instances.

Observation 2.2.2 *Every instance of the fair allocation problem admits an α-MMS allocation if this holds for all α-irreducible instances.*

The reducibility argument plays an important role in both the existential proofs and algorithms.

2.3 Our Results and Techniques

Throughout this chapter, we study the fair allocation problem for additive and non-additive agents. Kurokawa *et al.* [2] study the fair allocation problem and show a $2/3$-MMS allocation is guaranteed to exist for any number of additive agents. We improve this result in two different dimensions: (i) we improve the factor $2/3$ to a factor $3/4$ for additive agents. (ii) we provide similar guarantees for submodular, fractionally subadditive, and subadditive agents. Moreover, we provide algorithms that find such allocations in polynomial time. A brief summary of our results is illustrated in Table 2.1.

2.3.0.1 Additive Setting

As mentioned before, the pioneering work of Kurokawa *et al.* [2] present the first proof to the existence of a $2/3$-MMS allocation in the additive setting. On the negative side, they show that their analysis is tight, i.e. their method cannot be used to obtain a better approximation guarantee. However, whether or not a better bound could be achieved via a more efficient algorithm remains open as Kurokawa *et al.* [2] pose it as an open problem.

We answer the above question in the affirmative. Our main contribution is a proof to the existence of a $3/4$-MMS allocation for additive agents. Furthermore, we show that such an allocation can be found in polynomial time.

Theorem 2.3.1 *Any fair allocation problem with additive agents admits a* $3/4$-MMS *allocation. Moreover, a* $(3/4-\epsilon)$-MMS *allocation can be found in time* $\mathsf{poly}(n, m)$ *for any*

Previous work	Additive	Submodular	XOS	Subadditive
Existential proof	$2/3$ [2]	$1/10$ [19][a]	-	-
Polytime algorithm	$2/3-\epsilon$ [12]	$1/31$ [19]	-	-
Upper bound	$1-\epsilon$ [2]	-	-	-

Our results	Additive	Submodular	XOS	Subadditive
Existential proof	$3/4$	$1/3$	$1/5$	$1/10\lceil\log m\rceil$
Polytime algorithm	$3/4-\epsilon$	$1/3$	$1/8$	-
Upper bound	-	$3/4$	$1/2$	$1/2$

Table 2.1: Summary of the results

[a] In a parallel work to ours, Barman and Murthy in [19] (**EC'17**) consider the submodular case and propose a $1/31$ approximation guarantee.

$\epsilon > 0$.

The above theorem is surprising, since most of the previous methods provided for proving the existence of a $2/3$-MMS allocation were tight. This convinced many in the community that $2/3$ is the best that can be guaranteed. This shows that the current techniques and known structural properties of maximin share are not powerful enough to prove the bounds better than $2/3$. In this chapter, we provide a better understanding of this notion by demonstrating several new properties of maximin share. For example, we introduce a generalized form of reducibility and develop double counting techniques that are closely related to the concept of maximin-share.

For a better understanding of our algorithm, we start with the case where valuations of the agents for all items are small enough. More precisely, let $0 < \alpha < 1$ be a constant number and assume for every agent a_i and every item b_j, the value of agent a_i for item b_j is bounded by $\alpha\mathsf{MMS}_i$. In this case, we propose the following simple procedure to allocate the items to the agents.

- Arrange the items in an arbitrary order.

- Start with an empty bag and add the items to the bag one by one with respect to their order.

- Every time the valuation of an agent a_i for the set of items in the bag reaches $(1-\alpha)\mathsf{MMS}_i$, give all items of the bag to that agent, and continue with an empty bag. In case many agents are qualified to receive the items, we choose one of them arbitrarily. From this point on, we exclude the agent who received the items from the process.

We call this procedure the bag filling algorithm. One can see this algorithm as an extension of the famous moving knife algorithm for indivisible items. It is not hard to show that the bag filling algorithm guarantees a $(1-\alpha)$-MMS allocation to all of the agents. The crux of the argument is to show that every agent receives at least one bag of items. To this end, one could argue that every time a set of items is allocated to an agent a_i, no other agent a_j loses a value more than MMS_j. This, together with the fact that $V_i(\mathcal{M}) \geq n\mathsf{MMS}_i$, shows that at the end of the algorithm, every agent receives a fair share ($(1-\alpha)$-MMS) of the items.

This observation sheds light on the fact that low-value items can be distributed in a more efficient way. Therefore, the main hardness is to allocate the items with higher values to the agents. To overcome this hardness, we devise a clustering method. Roughly speaking, we divide the agents into three clusters according to their valuation functions. We prove desirable properties for the agents of each cluster. Finally, via a procedure that is similar in spirit to the bag filling algorithm but more complicated, we allocate the items to the agents.

Our clustering method is based on three important principles: *reducibility*, *matching allocation*, and *cycle-envy-freeness*. We give a brief description of each principle in the following.

Reducibility: The reducibility principle is very simple and elegant but plays an important role in the allocation process. Roughly speaking, consider a situation where for an agent a_i and a set S of items we have the following properties: $V_i(S) \geq \alpha \mathsf{MMS}_i$, and for all $a_j \neq a_i$, $\mathsf{MMS}_j^{n-1}(\mathcal{M} \setminus S) \geq \mathsf{MMS}_j$, where $V_i(S)$ is the valuation of agent a_i for subset S of items. Intuitively, since the maxmin shares of all agents except a_i for all the items other than set S are at least as much as their current maxmin shares, allocating set S to a_i cannot hurt the guarantee. In other words, given that an α-MMS allocation is possible for all agents except a_i with items not in S, we can allocate set S to agent a_i and recursively solve the problem for the rest of the agents. Although the definition of reducibility is more general than what mentioned above, the key idea is that reducible instances of the problem can be transformed into irreducible instances (see Observation 2.2.2). This makes the problem substantially simpler, since α-irreducible instances of the problem have many desirable properties. For example, in such instances, the value of every agent a_i for each item is less than $\alpha \mathsf{MMS}_i$. By setting $\alpha = 1/2$, this observation along with the analysis of the $\mathsf{bag\ filling}$ algorithm, proves the existence of a $1/2$-MMS allocation. It is worth to mention that a special form of reducibility, where $|S| = 1$ is used in the previous works [12, 2].

Matching allocation: At the core of the clustering part, we use a well-structured type of matching to allocate the items to the agents. Intuitively, we cluster the agents to deal with high-value or in other words *heavy* items. In order to cluster a group of agents, we find a subset T of agents and a subset S of items, together with a matching M from S to T. We choose T, S, and M in a way that (i) every item assigned to an agent has a value of at least β to him, (ii) agents who do not receive any items have a value less than β for each of the assigned items. Such an allocation requires careful application of several properties of maximal matchings in bipartite graphs. A matching with similar structural properties is previously used by Kurokawa *et al.* [2] to allocate the bundles to the agents.

Cycle-envy-freeness: Envy-freeness is itself a well-known notion for fairness in the resource allocation problems. However, this notion is perhaps more applicable to the allocation of divisible goods. In our algorithm, we use a much weaker notion of envy-freeness, namely *cycle-envy-freeness*. A cycle-envy-free allocation contains no cyclic permutation of agents, such that each agent envies the next agent in the cycle. In the clustering phase, we choose a matching M in a way that preserves cycle-envy-freeness for the clustered agents.

Cycle-envy-freeness plays a key role in the second phase of the algorithm. As aforementioned, our method in the assignment phase is closely related to the bag filling procedure described above. The difference is that the efficiency of our method depends on the order of the agents who receive the items. Based on the notion of cycle-envy-freeness, we prioritize the agents and, as such, we show the allocation is fair. An analogous concept is previously used in [31], albeit with a different application than ours.

2.3.0.2 Submodular, XOS, and Subadditive Agents

Although the problem was initially proposed for additive agents, it is very well-motivated to extend the definition to other classes of set functions. For instance, it is quite natural to expect that an agent prefers to receive two items of value 400, rather than receiving 1000 items of value 1. Such a constraint cannot be imposed in the additive setting. However, submodular functions which encompass k-demand valuations are strong tools for modeling these constraints. Such generalizations have been made to many similar problems, including the *Santa Claus max-min fair allocation*, *welfare maximization*, and *secretary* problems [32, 26, 27, 33]. The most common classes of set functions that have been studied before are submodular, XOS, and subadditive functions. We consider the fair allocation problem when the agents' valuations are in each of these classes. In contrast to the additive setting in which finding a constant MMS allocation is trivial, the problem becomes much more subtle even when the agents' valuations are *monotone submodular*. For instance, the bag filling algorithm does not promise any constant approximation factor for submodular agents, while it is straight-forward to show it guarantees a $(1-\alpha)$-MMS allocation for additive agents.

We begin with submodular set functions. We show that the fair allocation problem with submodular agents admits a $1/3$-MMS allocation. In addition, we show, given access to *query oracles*, one can find such an allocation in polynomial time. We further complement our result by showing that a $3/4$-MMS is the best guarantee that one can hope to achieve in this setting. This is in contrast to the additive setting for which the only upper bound is that 1-MMS allocation is not always possible. We begin by stating an existential proof.

Theorem 2.3.2 *The fair allocation problem with submodular agents admits a* $1/3$-MMS *allocation.*

Our proof for submodular agents is fundamentally different from that of the additive setting. First, without loss of generality, we assume $\mathsf{MMS}_i = 1$ for every agent $a_i \in \mathcal{N}$. Moreover, we assume the problem is $1/3$-irreducible since otherwise we can reduce the problem. Next, given a function $f(.)$, we define the *ceiling function* $f^x(.)$ as follows:

$$f^x(S) = \min\{x, f(S)\} \qquad \forall S \subseteq \mathsf{ground}(f).$$

An important property of the ceiling functions is that they preserve submodularity, fractionally subadditivity, and subadditivity. We define the bounded welfare of an allocation $\mathcal{A}$ as $\sum V_i^{2/3}(A_i)$. Given that, we show an allocation that maximizes the bounded welfare is $1/3$-MMS. To this end, let $\mathcal{A}$ be an allocation with the maximum bounded welfare and suppose for the sake of contradiction that in such an allocation, an agent a_i receives a bundle of worth less than $1/3$ to him. Since $\mathsf{MMS}_i = 1$, agent a_i can divide the items into n sets, where each set is of worth at least 1 to him. For a valuation function V, define the contribution of an item b_j in set S ($b_j \in S$) as $V(S) - V(S \setminus \{b_j\})$. Now, we randomly select an element b_j which is *not* allocated to a_i. By the properties of submodular functions, we show that if we allocate b_j to a_i, the expected contribution of b_j to the bounded valuation function of a_i would be more than the current expected contribution of b_j to the bounded welfare of the allocation. Therefore, there exists an item b_j such that if we allocate that item to agent a_i, the total bounded welfare of the allocation will be increased. This contradicts the maximality of the allocation.

Notice that Theorem 2.3.2 is only an existential proof. A natural approach to find such a solution is to start with an arbitrary allocation and iteratively increase its bounded welfare until it becomes $1/3$-MMS. The main challenge though is that we do not even know what the MMS values are. Furthermore, unlike the additive setting, we do not have any PTAS algorithm that provides us a close estimate to these values. To overcome this challenge, we propose a combinatorial trick to guess these values without incurring any additional factor to our guarantee. The high level idea is to start with large numbers as estimates to the MMS values. Every time we run the algorithm on the estimated values, it either finds a desired allocation, or reports that the maximin value of an agent is misrepresented by at least a multiplicative factor. Given this, we divide the maximin value of that agent by that factor and continue on with the new estimates. Therefore, at every step of the algorithm, we are guaranteed that our estimates are not less than the actual MMS values. Based on this, we show that the running time of the algorithm is polynomial, and that the resulting allocation has the desired properties.

Theorem 2.3.3 *Given access to query oracles, one can find a* $1/3$-MMS *allocation to submodular agents in polynomial time.*

Finally, we show that in some instances with submodular agents, no allocation is better than $3/4$-MMS.

Theorem 2.3.4 *For any integer number* $c > 0$*, there exists an instance of the fair allocation problem with* $n \geq c$ *submodular agents for which no allocation is better than*

$3/4$-MMS.

We show Theorem 2.3.4 by a counter-example. In this counter-example we have n agents and $2n$ items. Moreover, the valuation functions of the first $n-1$ agents are the same, but the last agent has a slightly different valuation function that makes it impossible to find an allocation which is better than $3/4$-MMS. The number of agents in this example can be arbitrarily large.

Similar to the submodular setting, we provide an upper bound on the quality of any allocation in the XOS setting. We show the following theorem by a counter-example.

Theorem 2.3.5 *For any integer number c, there is an instance of the fair allocation problem with XOS agents where $n \geq c$ and no allocation is better than* $1/2$-MMS.

Next, we state the main theorem in the XOS setting.

Theorem 2.3.6 *The fair allocation problem with XOS agents admits a* $1/5$-MMS *allocation.*

Our approach for proving Theorem 2.3.6 is similar to the proof of Theorem 2.3.2. Again, we scale the valuations to make sure $\mathsf{MMS}_i = 1$ all agents and define the notion of bounded welfare, but this time as $\sum V_i^{2/5}(A_i)$. However, as XOS functions do not adhere to the nice structure of submodular functions, we use a different analysis to prove this theorem. Let $\mathcal{A}$ be an allocation with the maximum bounded welfare. In case all agents receive a value of at least $1/5$, the proof is complete. Otherwise, let a_i be an agent that receives a set of items whose value to him is less than $1/5$. In contrast to the submodular setting, giving no item alone to a_i can guarantee an increase in the bounded welfare of the allocation. However, this time, we show there exists a set S of items such that if we take them back from their recipients and instead allocate them to agent a_i, the bounded welfare of the allocation increases. The reason this holds is the following: since $\mathsf{MMS}_i = 1$, agent a_i can split the items into $2n$ sets where every set is worth at least $2/5$ to a_i, otherwise the problem is $1/5$-reducible. Moreover, since the valuation functions are XOS, we show that giving one of these $2n$ sets to a_i will increase the bounded welfare of the allocation. Therefore, if $\mathcal{A}$ is maximal, then $\mathcal{A}$ is also $1/5$-MMS.

Finally, we show that a $1/8$-MMS allocation in the XOS setting can be found in polynomial time. Our algorithm only requires access to demand and XOS oracles. Note that this bound is slightly worse than our existential proof due to some computational hardnesses. However, the blueprint of the algorithm is based on the proof of Theorem 2.3.6.

Theorem 2.3.7 *Given access to demand and XOS oracles, we can find a* $1/8$-MMS *allocation for the problem with XOS agents in polynomial time.*

We start with an arbitrary allocation and increase the bounded welfare until the allocation becomes $1/8$-MMS. The catch is that if the allocation is not $1/8$-MMS, then there exists an agent a_i and a set S of items such that if we take back these items from their current recipients and allocate them to agent a_i, the bounded welfare of the allocation increases. In order to increase the bounded welfare, there are two computational barriers that need to be lifted. First, similar to the submodular setting, we do not have any estimates to the MMS values. Analogously, we resolve the first issue by iteratively guessing the MMS values. The second issue is that in every step of the algorithm, we have to find a set S of items to allocate to an agent a_i that results in an increase in the bounded welfare. Such a set S cannot be trivially found in polynomial time. That is where the demand and XOS oracles take part. We show how to find such a set in polynomial time. The high-level idea is the following: first, by accessing the XOS oracles, we determine the contribution of every item to the bounded welfare of the allocation. Next, we set the price of every element equal to three times the contribution of that element to the bounded welfare and run the demand oracle to find which subset has the highest profit for agent a_i. We show this subset has a value of at least $1/4$ to a_i. Next, we sort the elements of this set based on the ratio of contribution to the overall value of the set over the price of the item, and select a prefix for them that has a value of at least $1/4$ to a_i. Finally, we argue that allocating this set to a_i increases the bounded welfare of the allocation by at least some known lower bound. This, married with the combinatorial trick to guess the MMS values, gives us a polynomial time algorithm to find a $1/8$-MMS allocation.

An immediate corollary of Theorems 2.3.3 and 2.3.7 is a polynomial time algorithm for approximating the maxmin value of a submodular and an XOS function within factors $1/3$ and $1/8$, respectively.

Corollary 2.3.8 *Let f be a submodular/XOS function on a set of ground elements S, and let n be an integer number. Given access to query oracle/demand and XOS oracles of f, we can partition the elements of S into n disjoint subsets $S_1, S_2, \ldots, S_n$ such that*

$$\min_{i=1}^{n} f(S_i) \geq c \cdot \mathsf{MMS}_f^n$$

where MMS_f^n *denotes the maxmin value for function* f *on* n *subsets. Constant* c *equals* $1/3$ *if* f *is submodular and is equal to* $1/8$ *for the XOS case.*

Finally, we investigate the problem when the agents are subadditive and present an existential proof based on a well-known reduction to the XOS setting.

Theorem 2.3.9 *The fair allocation problem with subadditive agents admits a* $1/10\lceil \log m \rceil$-MMS *allocation.*

2.4 A Brief Overview of the $3/4$-MMS Algorithm

The purpose of this section is to present an abstract overview over the ideas behind our algorithm for finding a $3/4$-MMS allocation in the additive setting. For simplicity, we start with a simple $1/2$-MMS algorithm mentioned in Section 3.1.1. Recall that the bag filling procedure guarantees a $1-\alpha$ approximation solution when the valuations of the agents for each item is smaller than α. Furthermore, we know that in every α-irreducible instance, all the agents have a value less than α for every item. Thus, the following simple procedure yields a $1/2$-MMS allocation:

1. Reduce the problem until no agent has a value more than $1/2$ for any item.

2. Allocate the items to the agents via a bag filling procedure.

Figure 2.1 shows a schematic representation of this algorithm. We can extend the idea in $1/2$-MMS algorithm to obtain a more efficient algorithm. Here is the sketch of the $2/3$-MMS algorithm: consider a $2/3$-irreducible instance of the problem. In this instance, we have no item with a value more than or equal to $2/3$ to any agent. Nevertheless, the

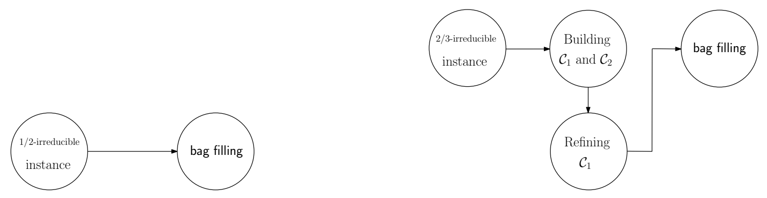


Figure 2.2: $2/3$-MMS Algorithm

items are not yet small enough to run a bag filling procedure. The idea here is to divide the agents into two clusters $\mathcal{C}_1$ and $\mathcal{C}_2$. Along this clustering, the items with a value in range $[1/3, 2/3)$ are given to the agents. In particular, one item is allocated to every agent in $\mathcal{C}_1$ that is worth at least $1/3$ to him and less than $1/3$ to any agent not in $\mathcal{C}_1$. Next, we refine Cluster $\mathcal{C}_1$. In the refining procedure, if any remaining item could singly satisfy an agent in $\mathcal{C}_1$, we do so. After building $\mathcal{C}_1$ and $\mathcal{C}_2$ and refining $\mathcal{C}_1$, the remaining items preserve the following two invariants:

1. The value of every remaining item is less than $1/3$ to every remaining agent.

2. No remaining item can singly satisfy an agent in $\mathcal{C}_1$ (regarding the item that is already allocated to them)

These two invariants enable us to run a bag filling procedure over the remaining items. For this case, the bag filling procedure must be more intelligent: in the case that multiple agents are qualified to receive the items of the bag, we prioritize the agents. Roughly speaking, the priorities are determined by two factors: the cluster they belong to, and the cycle-envy-freeness property of the agents in $\mathcal{C}_1$. In Figure 2.2 you can see a flowchart for this algorithm.

Our method for a $3/4$-MMS allocation takes one step further from the previous $2/3$-MMS algorithm. Again, we assume that the input is $3/4$-irreducible since otherwise it can

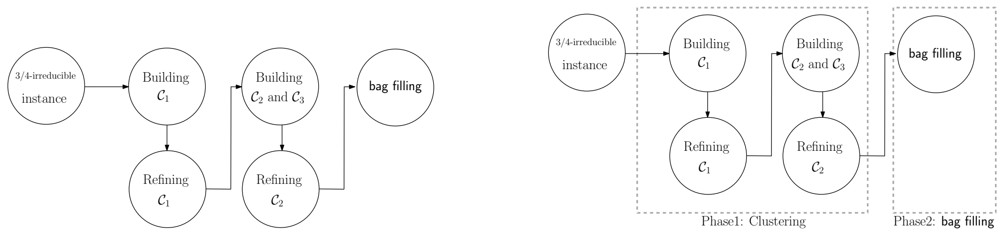


Figure 2.4: Algorithm Phases

be further simplified. Via similar ideas, we build Cluster $\mathcal{C}_1$ and refine it. Next, we build Clusters $\mathcal{C}_2$ and $\mathcal{C}_3$ and refine $\mathcal{C}_2$. After refining Cluster $\mathcal{C}_2$, the following invariants are preserved for the remaining items:

1. Almost every remaining item has a value less than $1/4$ to every remaining agent. More precisely, for every remaining agent a_i, there is at most one remaining item b_j with $V_i(\{b_j\}) \geq 1/4$.

2. No remaining item can singly satisfy an agent in $\mathcal{C}_1$ and $\mathcal{C}_2$ (regarding the item that is already allocated to them).

Finally, we run a bag filling procedure. Again, in the bag filling procedure, the priorities of the agents are determined by the cluster they belong to, and the cycle-envy-freeness of the clusters. In Figure 2.3, a flowchart for this algorithm is shown. Our assumption is that the input is $3/4$-irreducible. Hence, we describe our algorithm in two phases: a clustering phase and the bag filling phase, as shown in Figure 2.4.

We show that all the steps of the algorithm can be implemented in polynomial time. Furthermore, we show that the assumption that the input is $3/4$-irreducible is without loss of generality. In fact, we show that it suffices to check some invariants of irreducibility to be held in certain points of the algorithm.

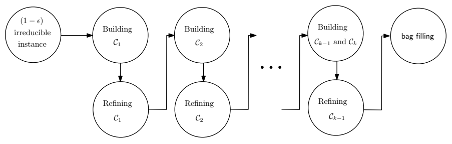


Figure 2.5: Generalizing the algorithm into k clusters

2.5 Additive Agents

In this section we study the fair allocation problem in the additive setting. We present a proof to the existence of a $3/4$-MMS allocation when the agents are additive. This improves upon the work of Kurokawa, Procaccia, and Wang [2] wherein the authors prove a $2/3$-MMS allocation exists for any combination of additive agents. As we show, our proof is constructive; given an algorithm that determines the MMS of an additive set function within a factor α, we can implement an algorithm that finds a $3/(4\alpha)$-MMS allocation in polynomial time. This married with the PTAS algorithm of Epstein and Levin [34] for finding the MMS values, results is an algorithm that finds a $3/(4+\epsilon)$-MMS allocation in polynomial time.

The main idea behind the $3/4$-MMS allocation is *clustering* the agents. Roughly speaking, we categorize the agents into three clusters, namely $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. We show that the valuation functions of the agents within each cluster show similar behaviors. Along the clustering process, we allocate the heavy items (the items that have a valuation of at least $1/4$ to some agents) to the agents. By Observation 2.2.2, proving a $3/4$-MMS guarantee can be narrowed down to only $3/4$-irreducible instances. The $3/4$-irreducibility of the problem guarantees that after the clustering process, the remaining items are light. This enables us to run a bag filling process to satisfy the agents. In order to prove the correctness of the algorithm, we take advantage of the properties of each cluster separately.

The organization of this section is summarized in the following: we start by a brief and abstract explanation of the ideas in Section 2.4. In Section 2.5.1 we study the properties of the additive setting and state the main observations that later imply the correctness of our algorithm. Next, in Section 2.5.2 we discuss a method for clustering the agents and in Section 2.5.3 we show how we allocate the items to the agents of each cluster to ensure a $3/4$-MMS guarantee. Finally, in Section 2.5.5 we explain the implementation details and prove a polynomial running time for the proposed algorithm.

Throughout this section, we assume $\mathsf{MMS}_i = 1$ for all agents $a_i \in \mathcal{N}$. This is without loss of generality for the existential proof since one can scale the valuation functions to impose this constraint. However, the computational complexity of the allocation will be affected by this assumption since determining the MMS of an additive function is NP-hard [34]. That said, we show in Section 2.5.5 that this challenge can be overcome by incurring an additional $1 + \epsilon$ factor to the approximation guarantee.

2.5.1 General Definitions and Observations

Throughout this section we explore the properties of the fair allocation problem with additive agents.

2.5.1.1 Consequences of Irreducibility

Since the objective is to prove the existence of a $3/4$-MMS allocation, by Observation 2.2.2, it only suffices to show every $3/4$-irreducible instance of the problem admits a $3/4$-MMS allocation. Therefore, in this section we provide several properties of the $3/4$-irreducible instances. We say a set S of items *satisfies* an agent a_i if and only if $V_i(S) \geq 3/4$. Perhaps the most important consequence of irreducibility is a bound on the valuation of the agents for every item. In the following we show if the problem is $3/4$-irreducible, then no agent has a value of $3/4$ or more for an item.

Lemma 2.5.1 *For every α-irreducible instance of the problem we have*

$$\forall a_i \in \mathcal{N}, b_j \in \mathcal{M} \qquad V_i(b_j) < \alpha.$$

In other words, Lemma 2.5.1 states that in a $3/4$-irreducible instance of the problem, no item alone can satisfy an agent.

It is worth mentioning that the proof for Lemma 2.5.1 does not rely on additivity of the valuation functions and holds as long as the valuations are monotone. Thus, regardless of the type of the valuation functions, one can assume that in any α-irreducible instance, value of any item is less than α for any agent. Hence the statement carries over to the submodular, XOS, and subadditive settings.

Proof of Lemma 2.5.1: The key idea is that given $\mathsf{MMS}_i \geq 1$ for an agent a_i, then for every item $b_j \in \mathcal{M}$ we have $\mathsf{MMS}_i^{n-1}(\mathcal{M} \setminus b_j) \geq 1$. This holds since removing an item from $\mathcal{M}$ will diminish the value of at most one partition in the optimal n partitioning of the items. Therefore, at least $n-1$ partitions have a value of 1 or more to a_i and thus $\mathsf{MMS}_i^{n-1}(\mathcal{M} \setminus b_j) \geq 1$. The rest of the proof follows from the definition of α-irreducibility. If the valuation of an item b_j to an agent a_i is at least α, then the problem is α-reducible since if we allocate b_j to a_i, we have

$$\mathsf{MMS}_{V_k}^{n-1}(\mathcal{M} \setminus \{b_j\}) \geq 1$$

for every agent $a_k \neq a_i$. This contradicts with the α-irreducibility assumption. $\square$

As a natural generalization of Lemma 2.5.1, we show a similar observation for every pair of items. However, this involves an additional constraint on the valuation of the other agents for the pertinent items. In contrast to Lemma 2.5.1, Lemmas 2.5.2 and 2.5.3 are restricted to additive setting and their results do not hold in more general settings.

Lemma 2.5.2 *If the problem is $3/4$-irreducible and $V_i(\{b_j, b_k\}) \geq 3/4$ holds for an agent $a_i \in \mathcal{N}$ and items $b_j, b_k \in \mathcal{M}$, then there exists an agent $a_{i'} \neq a_i$ such that*

$$V_{i'}(\{b_j, b_k\}) > 1$$

Proof. Suppose for the sake of contradiction that for every agent $a_{i'} \neq a_i$ we have $V_{i'}(\{b_j, b_k\}) \leq 1$. By this assumption, we show

$$\mathsf{MMS}_{i'}^{n-1}(\mathcal{M} \setminus \{b_j, b_k\}) \geq 1 \tag{2.1}$$

holds. This is true since removing two items b_j and b_k from $\mathcal{M}$ decreases the value of at most two partitions of the optimal partitioning of $\mathcal{M}$ for $\mathsf{MMS}_{i'}$. If $n-1$ partitions remain intact, then Inequality (2.1) trivially holds. If not, merging the two partitions that initially contained b_j and b_k results in a partition with value at least 1 to a_i. This partition together with the $n-2$ remaining partitions result in a desirable partitioning of $\mathcal{M}$ into $n-1$ partitions. Therefore, Inequality (2.1) holds for any agent $a_{i'}$, and this implies that by allocating $S = \{b_j, b_k\}$ to a_i, not only does $V_i(S) \geq 3/4$ hold, but also for every $a_{i'} \neq a_i$ we have

$$\mathsf{MMS}_{i'}^{n-1}(\mathcal{M} \setminus \{b_j, b_k\}) \geq 1$$

which means the problem is $3/4$-reducible, and it contradicts our assumption. $\square$

According to Lemma 2.5.2, in every $3/4$-irreducible instance of the problem, for every agent a_i and items b_j, b_k, either $V_i(\{b_j, b_k\}) < 3/4$ or there exists another agent $a_{i'} \neq a_i$, such that $V_{i'}(\{b_j, b_k\}) > 1$. Otherwise, we can reduce the problem and find a $3/4$-MMS allocation recursively. More generally, let $S = \{b_{j_1}, b_{j_2}, \ldots, b_{j_{|S|}}\}$ be a set of items in $\mathcal{M}$ and $T = \{a_{i_1}, a_{i_2}, \ldots, a_{i_{|T|}}\}$ be a set of agents such that

(i) $|S| = 2|T|$

(ii) For every $a_{i_a} \in T$ we have $V_{i_a}(\{b_{j_{2a-1}}, b_{j_{2a}}\}) \geq 3/4$.

(iii) For every $a_i \notin T$ we have $V_i(\{b_{j_{2a-1}}, b_{j_{2a}}\}) \leq 1$ for every $1 \leq a \leq |T|$.

then the problem is $3/4$-reducible.

Lemma 2.5.3 *In every $3/4$-irreducible instance of the problem, for every set $T = \{a_{i_1}, a_{i_2}, \ldots, a_{i_{|T|}}\}$ of agents and set $S = \{b_{j_1}, b_{j_2}, \ldots, b_{j_{|S|}}\}$ of items at least one of the above conditions is violated.*

Proof. The proof for this lemma is obtained by applying Lemma 2.5.2, $|T|$ times. Consider an agent $a_i \notin T$. According to the argument in Lemma 2.5.2, if we assign b_{j_1} and

b_{j_2} to a_{i_1}, a_i can partition the items in $\mathcal{M} \setminus \{b_{j_1}\ b_{j_2}\}$ into $n-1$ partitions with value at least 1 to a_i, i.e.

$$\mathsf{MMS}_i^{n-1}(\mathcal{M} \setminus \{b_{j_1}, b_{j_2}\}) \geq 1.$$

By the same deduction, after assigning b_{j_3} and b_{j_4} to a_{i_2}, we have

$$\mathsf{MMS}_i^{n-2}(\mathcal{M} \setminus \{b_{j_1}, b_{j_2}, b_{j_3}, b_{j_4}\}) \geq 1.$$

By repeating above argument $|T|$ times, we have:

$$\mathsf{MMS}_i^{n-|T|}(\mathcal{M} \setminus S) \geq 1.$$

On the other hand, by condition (II), every agent a_{i_k} satisfies with items $b_{j_{2k-1}}$ and $b_{j_{2k}}$. This means that we can reduce the instance by satisfying the agents in T by the items in S, which is a contradiction by the irreducibility assumption. □

2.5.1.2 Modeling the Problem with Bipartite Graphs

In our algorithm we subsequently make use of classic algorithms for bipartite graphs. Let $G = \langle V(G), E(G) \rangle$ be a graph representing the agents and the items. Moreover, let $V(G) = \mathcal{X} \cup \mathcal{Y}$ where $\mathcal{Y}$ corresponds to the agents and $\mathcal{X}$ corresponds to the items. More precisely, for every agent a_i we have a vertex $y_i \in \mathcal{Y}$ and every item b_j corresponds to a vertex $x_j \in \mathcal{X}$. For every pair of vertices $y_i \in \mathcal{Y}$ and $x_j \in \mathcal{X}$, there exists an edge $(x_j, y_i) \in E(G)$ with weight $w(x_j, y_i) = V_i(\{b_j\})$. We refer to this graph as *the value graph*.

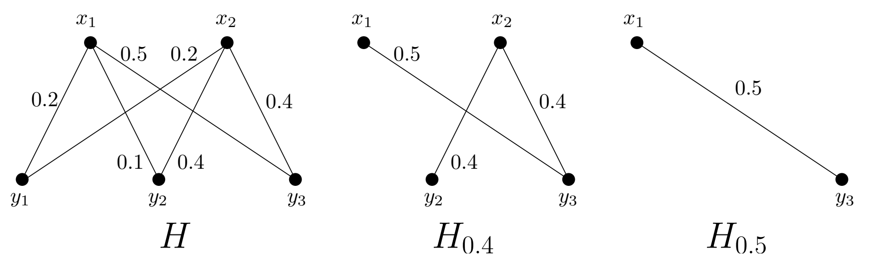


Figure 2.6: An example of β-filtering on a graph. After removing the edges with a value smaller than β, some vertices may become isolated. All such vertices are removed from the filtered graph.

We define an operation on the weighted graphs which we call *filtering*. Roughly speaking, a filtering is an operation that receives a weighted graph as input and removes all of the edges with weight less than a threshold from the graph. Next, we remove all of the isolated[2] vertices and report the remaining as the filtered graph. In the following we formally define the notion of filtering for weighted graphs.

Definition 2.5.4 *A β-filtering of a weighted graph $H\langle V(H), E(H)\rangle$, denoted by $H_\beta\langle V_\beta(H), E_\beta(H)\rangle$, is a subgraph of H where $V_\beta(H)$ is the set of all vertices in $V(H)$ incident to at least one edge of weight β or more and*

$$E_\beta(H) = \{(u,v) \in E(H) | w(u,v) \geq \beta\}.$$

For the case of the value graph, we also denote by $\mathcal{Y}_\beta$ and $\mathcal{X}_\beta$ the sets of agents and items corresponding to vertices of $V_\beta(G)$. Figure 2.6 illustrates an example of a graph H, together with $H_{0.4}$ and $H_{0.5}$. Note that none of the vertices in $H_{0.4}$ or $H_{0.5}$ are isolated.

Denote by a maximum matching, a matching that has the highest number of edges in a graph. In definition 2.5.5, we introduce our main tool for clustering the agents.

[2]A vertex is called isolated if no edge is incident to that vertex.

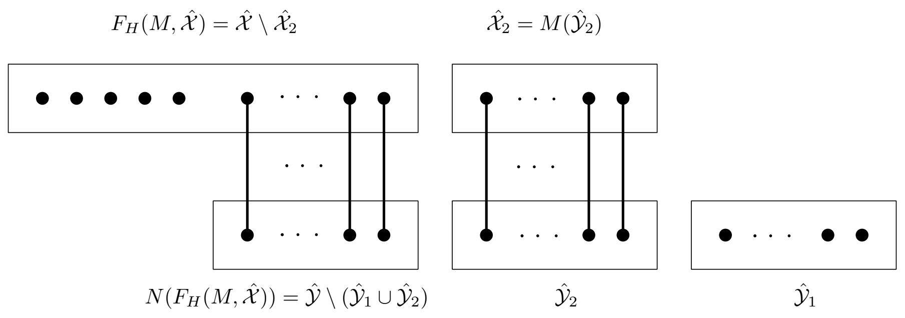

$F_H(M, \hat{\mathcal{X}}) = \hat{\mathcal{X}} \setminus \hat{\mathcal{X}}_2$
$\hat{\mathcal{X}}_2 = M(\hat{\mathcal{Y}}_2)$
$N(F_H(M, \hat{\mathcal{X}})) = \hat{\mathcal{Y}} \setminus (\hat{\mathcal{Y}}_1 \cup \hat{\mathcal{Y}}_2)$
$\hat{\mathcal{Y}}_2$
$\hat{\mathcal{Y}}_1$

Figure 2.7: Definition of F_H

Definition 2.5.5 *Let $H\langle V(H), E(H)\rangle$ be a bipartite graph with $V(H) = \hat{\mathcal{X}} \cup \hat{\mathcal{Y}}$ and let M be a maximum matching of H. Define $\hat{\mathcal{Y}}_1$ as the set of the vertices in $\hat{\mathcal{Y}}$ that are not saturated by M. Also, define $\hat{\mathcal{Y}}_2$ as the set of vertices in $\hat{\mathcal{Y}}$ that are connected to $\hat{\mathcal{Y}}_1$ by an alternating path and let $\hat{\mathcal{X}}_2 = M(\hat{\mathcal{Y}}_2)$, where $M(\hat{\mathcal{Y}}_2)$ is the set of vertices in $\hat{\mathcal{X}}$ that are matched with the vertices of $\hat{\mathcal{Y}}_2$ in M. We define $F_H(M, \hat{\mathcal{X}})$ as the set of the vertices in $\hat{\mathcal{X}} \setminus \hat{\mathcal{X}}_2$.*

For a better understanding of Definition 2.5.5, consider Figure 2.7. By the definition of alternating paths, there is no edge between the saturated vertices of $F_H(M, \hat{\mathcal{X}})$ and $\hat{\mathcal{Y}}_1 \cup \hat{\mathcal{Y}}_2$. On the other hand, since M is maximum, the graph doesn't have any augmenting path. Thus, there is no edge between unsaturated vertices in $F_H(M, \hat{\mathcal{X}})$ and $\hat{\mathcal{Y}}_1 \cup \hat{\mathcal{Y}}_2$. As a result, there is no edge between $F_H(M, \hat{\mathcal{X}})$ and $\hat{\mathcal{Y}}_1 \cup \hat{\mathcal{Y}}_2$. Furthermore, $F_H(M, \hat{\mathcal{X}})$ has another important property: there exists a matching from $N(F_H(M, \hat{\mathcal{X}}))$ to $F_H(M, \hat{\mathcal{X}})$, that saturates all the vertices in $N(F_H(M, \hat{\mathcal{X}}))$, where $N(F_H(M, \hat{\mathcal{X}}))$ is the set of neighbors of $F_H(M, \hat{\mathcal{X}})$.

In Lemmas 2.5.7 and 2.5.6, we prove two remarkable properties for bipartite graphs. As a consequence of these two lemmas, Corollary 2.5.8 holds for every bipartite graph. We leverage the result of Corollary 2.5.8 in the clustering phase.

Lemma 2.5.6 *Let $H(V, E)$ be a bipartite graph with $V = \hat{\mathcal{X}} \cup \hat{\mathcal{Y}}$ and let M be a maximum matching of H. Then, for every set $T \subseteq \hat{\mathcal{X}} \setminus F_H(M, \hat{\mathcal{X}})$ we have $|N(T)| > |T|$, where $N(T)$ is the set of neighbors of T.*

Proof. We define $\hat{\mathcal{Y}}_1$ as the set of vertices in $\hat{\mathcal{Y}}$ that are not saturated by M, and $\hat{\mathcal{Y}}_2$ as the set of vertices in $\hat{\mathcal{Y}}$ that are connected to $\hat{\mathcal{Y}}_1$ by an alternating path. Moreover, let $\hat{\mathcal{X}}_2 = M(\hat{\mathcal{Y}}_2)$. By definition, $F_H(M, \hat{\mathcal{X}}) = \hat{\mathcal{X}} \setminus \hat{\mathcal{X}}_2$ (See Figure 2.7). As discussed before, all the vertices in $\hat{\mathcal{X}}_2$ are saturated by M. Consequently, all the vertices of T are saturated by M and $|N(T)| \geq |T|$.

Let $M(T)$ be the set of vertices which are matched to the vertices of T in M. We know that every vertex of T is present in at least one of the alternating paths which connect $\hat{\mathcal{Y}}_1$ to $\hat{\mathcal{Y}}_2$. Let

$$P = \langle \hat{y}_0, \hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2, \ldots, \hat{x}_k, \hat{y}_k \rangle$$

be one of these paths that includes at least one of the vertices of T. Since P is an alternating path which connects $\hat{\mathcal{Y}}_1$ to $\hat{\mathcal{Y}}_2$, $\hat{y}_0 \in \hat{\mathcal{Y}}_1$ (see Figure 2.8). In addition, according to the definition of alternating path, every edge $(\hat{x}_j, \hat{y}_j)$ of P belongs to M and every edge $(\hat{x}_j, \hat{y}_{j-1})$ does not belong to M.

Let $\hat{x}_i$ be the first vertex of T that appears in P. We know that the edge $(\hat{x}_i, \hat{y}_{i-1})$ does not belong to M. On the other hand, since $\hat{x}_i$ is the first vertex of T in M, $\hat{x}_{i-1} \notin T$. Note that $\hat{y}_{i-1}$ does not belong to $M(T)$, since every vertex of $M(T)$ is matched with a vertex of T in M and $(\hat{x}_{i-1}, \hat{y}_{i-1})$ is in M. The fact that $\hat{y}_{i-1} \notin M(T)$ means $N(T)$ contains at least one vertex that is not in $M(T)$. Since all the vertices in $M(T)$ are in $N(T)$, $|N(T)| > |M(T)|$ and hence, $|N(T)| > |T|$. Suppose x_i is the first vertex of T

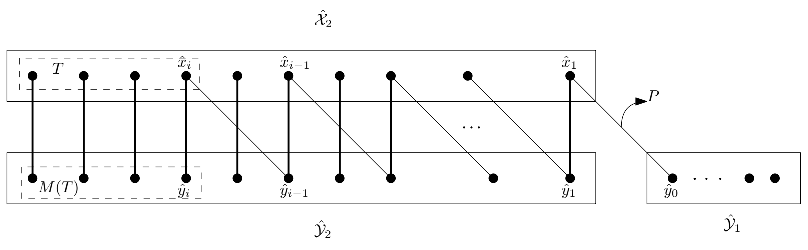


Figure 2.8: Alternating path P connects $\hat{\mathcal{Y}}_1$ to $\hat{\mathcal{Y}}_2$ and intersects T

in P. Since $x_1 \in \hat{\mathcal{Y}}_1$, $x_i \neq x_1$. By the assumption that x_i is the first vertex of T in P, x_{i-1} does not belong to T. In addition, x_{i-1} is connected to x_i, since (x_i, x_{i-1}) is an edge of P. Thus, the vertices in T have at least one neighbour, which is not in $M(T)$. Therefore, $N(T) = M(T)$ cannot hold, and we have $|N(T)| > |M(T)|$ which yields $|N(T)| > |T|$.

□

Lemma 2.5.7 *For a bipartite graph $H(V, E)$ with $V = \hat{\mathcal{X}} \cup \hat{\mathcal{Y}}$, $F_H(M, \hat{\mathcal{X}}) = \varnothing$ holds, if and only if for all $T \subseteq \hat{\mathcal{X}}$ we have $|N(T)| > |T|$, where $N(T)$ is the set of neighbors of T.*

Proof. If $F_H(M, \hat{\mathcal{X}}) = \varnothing$, according to Lemma 2.5.6,

$$\forall T \subseteq \hat{\mathcal{X}} \qquad |N(T)| > |T|.$$

On the other hand, suppose that for all $T \subseteq \hat{\mathcal{X}}$ we have $|N(T)| > |T|$. We show that $F_H(M, \hat{\mathcal{X}}) = \varnothing$. For the sake of contradiction, assume that $F_H(M, \hat{\mathcal{X}}) \neq \varnothing$ and let $T = F_H(M, \hat{\mathcal{X}})$. Since there exists a matching from T to $N(T)$ that saturates all the vertices of $N(T)$, we have $|T| \geq |N(T)|$, which is a contradiction. Hence, $F_H(M, \hat{\mathcal{X}}) = \varnothing$.

□

Corollary 2.5.8 (of Lemmas 2.5.7 and 2.5.6) *Let $H(V, E)$ be a bipartite graph with $V = \hat{\mathcal{X}} \cup \hat{\mathcal{Y}}$ and let M be a maximum matching of H. Furthermore, let $H'(V', E')$ be the induced sub-graph of H, with $V' = \hat{\mathcal{X}}' \cup \hat{\mathcal{Y}}'$, where $\hat{\mathcal{X}}' = \hat{\mathcal{X}} \setminus F_H(M, \hat{\mathcal{X}})$ and $\hat{\mathcal{Y}}' = \hat{\mathcal{Y}} \setminus N(F_H(M, \hat{\mathcal{X}}))$. Then, for any maximum matching M' of H', $F_{H'}(M', \hat{\mathcal{X}}') = \varnothing$ holds.*

2.5.1.3 Cycle-envy-freeness and MCMWM

In the algorithm, we satisfy each agent in two steps. More precisely, we allocate each agent two sets of items that are together of worth at least $3/4$ to him. We denote the first set of items allocated to agent a_i by f_i and the second set by g_i. Moreover, we attribute the agents with labels *satisfied*, *unsatisfied*, and *semi-satisfied* in the following way:

1. An agent a_i is satisfied if $V_i(f_i \cup g_i) \geq 3/4$.

2. An agent a_i is semi-satisfied if $f_i \neq \varnothing$ but $g_i = \varnothing$. In this case we define $\epsilon_i = 3/4 - V_i(f_i)$.

3. An agent a_i is unsatisfied if $f_i = g_i = \varnothing$.

As we see, the algorithm maintains the property that for every semi-satisfied agent a_i, $V_i(f_i) \geq 1/2$ holds and hence, $\epsilon_i < 1/4$.

To capture the competition between different agents, we define an attribution for an ordered pair of agents. We say a semi-satisfied agent envies another semi-satisfied agent, if he prefers to switch sets with the other agent.

Definition 2.5.9 *Let T be a set of semi-satisfied agents. An agent $a_i \in T$ envies an agent $a_j \in T$, if $V_i(f_j) \geq V_i(f_i)$. Also, we call an agent $a_i \in T$ a winner of T, if a_i envies no other agent in T. Similarly, we call an agent a_i a loser of T, if no other agent in T envies a_i.*

Note that it could be the case that an agent a_i is both a loser and a winner of a set T of agents. Based on Definition 2.5.9, we next define the notion of *cycle-envy-freeness*.

Definition 2.5.10 *We call a set T of semi-satisfied agents cycle-envy-free, if every non-empty subset of T contains at least one winner and one loser.*

Let C be a cycle-envy-free set of semi-satisfied agents. Define the representation graph of C as a digraph $G_C(V(G_C), \overrightarrow{E}(G_C))$, such that for any agent $a_i \in C$, there is a vertex v_i in $V(G_C)$ and there is a directed edge from v_i to v_j in $\overrightarrow{E}(G_C)$, if a_i envies a_j. In Lemma 2.5.11, we show that G_C is acyclic.

Lemma 2.5.11 *For every cycle-envy-free set of semi-satisfied agents C, G_C is a DAG.*

Proof. Consider a cycle L in G_C. For each vertex $v_j \in L$, there is at least one vertex $v_i \in L$ such that a_i envies a_j. Therefore, Considering S as the set of agents with vertices in L, none of the agents of S is a loser. By the same deduction, none of the agents of S is a winner. But this contradicts the fact that the set C is cycle-envy-free. $\square$

Definition 2.5.12 *A topological ordering of a cycle-envy-free set C of semi-satisfied agents, is a total order $\prec_O$ corresponding to the topological ordering of the representation graph G_C. More formally, for the agents $a_i, a_j \in C$ we have $a_i \prec_O a_j$ if and only if v_i appears before v_j, in the topological ordering of G_C.*

Note that in the topological ordering of a cycle-envy-free set C of semi-satisfied agents, if $a_i \in C$ envies $a_j \in C$, then $a_i \prec_O a_j$.

Observation 2.5.13 *Let C be a cycle-envy-free set of semi-satisfied agents. Then, for every agent $a_i \in C$ such that $a_j \prec_O a_i$, we have:*

$$V_i(f_j) \leq 3/4 - \epsilon_i.$$

We define a maximum cardinality maximum weighted matching of a weighted graph as a matching that has the highest number of edges and among them the one that has the highest total sum of edge weights. For brevity we call such a matching an MCMWM. In Lemma 2.5.14, we show that an MCMWM of a weighted bipartite graph has certain properties that makes it useful for building cycle-envy-free clusters.

Lemma 2.5.14 *Let $H\langle V(H), E(H)\rangle$ be a weighted bipartite graph with $V(H) = \hat{\mathcal{X}} \cup \hat{\mathcal{Y}}$ and let $M = \{(\hat{x}_1, \hat{y}_1), ..., (\hat{x}_k, \hat{y}_k)\}$ be an MCMWM of H. Then, for every subset $T \subseteq \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k\}$, the following conditions hold:*

- *There exists a vertex $\hat{y}_j \in T$ which is a winner in T, i.e., $w(\hat{x}_j, \hat{y}_j) \geq w(\hat{x}_i, \hat{y}_j)$, for all $\hat{x}_i \in M(T)$ and $(\hat{x}_i, \hat{y}_j) \in E(H)$.*
- *There exists a vertex $\hat{y}_j \in T$ which is a loser in T, i.e., $w(\hat{x}_i, \hat{y}_i) \geq w(\hat{x}_j, \hat{y}_i)$, for all $\hat{y}_i \in T$ and $(\hat{x}_j, \hat{y}_i) \in E(H)$.*
- *For any vertex $\hat{y}_i \in T$ and any unsaturated vertex $\hat{x}_j \in \hat{\mathcal{X}}$ such that $(\hat{x}_j, \hat{y}_i) \in E(H)$, $w(\hat{x}_i, \hat{y}_i) \geq w(\hat{x}_j, \hat{y}_i)$.*

where $M(T)$ is the set of vertices which are matched by the vertices of T in M.

Proof.

We describe the proof for the first condition in more details. The proof for the second condition is almost the same as the first condition.

The first condition: Suppose that there exists no such vertex. Our goal is to find a new matching of H with the same cardinality, but with more weight. To this end, we construct a directed graph H' from H as follows: for each $\hat{y}_j \in T$ we consider a vertex v_j in $V(H')$. Furthermore, there is a directed edge from v_j to v_i in H', if and only if $w(\hat{x}_j, \hat{y}_j) < w(\hat{x}_i, \hat{y}_j)$ in H.

If there exists a vertex v_j with out-degree zero in H', then $\hat{y}_j$ is the desired winner in T, since

$$\forall \hat{y}_i \in H, w(\hat{x}_j, \hat{y}_j) \geq w(\hat{x}_i, \hat{y}_j).$$

Otherwise, the out-degree of every vertex in T is non-zero. Therefore, H' has at least one cycle $L = \langle v_{l_1}, v_{l_2}, \ldots, v_{l_{|L|}} \rangle$. Now, if we change matching M by removing the set of edges

$$\{(\hat{y}_{l_1}, \hat{x}_{l_1}), (\hat{y}_{l_2}, \hat{x}_{l_2}), \ldots, (\hat{y}_{l_{|L|}}, \hat{x}_{l_{|L|}})\}$$

from M and adding

$$\{(\hat{y}_{l_1}, \hat{x}_{l_2}), (\hat{y}_{l_2}, \hat{x}_{l_3}), \ldots, (\hat{y}_{l_{|L|}}, \hat{x}_{l_1})\}$$

to M, the weight of our matching will be increased. Note that by the definition of an edge in H', we have

$$w(\hat{x}_{l_2}, \hat{y}_{l_1}) > w(\hat{x}_{l_1}, \hat{y}_{l_1}), w(\hat{x}_{l_3}, \hat{y}_{l_2}) > w(\hat{x}_{l_2}, \hat{y}_{l_2}), \ldots, w(\hat{x}_{l_1}, \hat{y}_{l_{|L|}}) > w(\hat{x}_{l_{|L|}}, \hat{y}_{l_{|L|}}).$$

But this contradicts the fact that M was MCMWM of H.

The second condition: The proof of this part is very similar to the proof of the first condition. For the sake of contradiction suppose that for every vertex $\hat{y}_j \in T$, there is at least one vertex $\hat{y}_i \in T$ where $w(\hat{x}_j, \hat{y}_i) > w(\hat{x}_i, \hat{y}_i)$ in H with $(\hat{x}_j, \hat{y}_i) \in E(H)$. Similar to the proof of the first condition, we construct a new directed graph H' from H where we have a vertex v_j in H' for each vertex $\hat{y}_j$ in T. For every pair $\hat{y}_i$ and $\hat{y}_j$ which are members of T we connect v_i to v_j with a directed edge in H' if

$$w(\hat{x}_j, \hat{y}_i) > w(\hat{x}_i, \hat{y}_i)$$

in H and $(\hat{x}_j, \hat{y}_i) \in E(H)$. Note that if H' contains a vertex v_i with in-degree equal to zero, then $\hat{y}_i$ is the desired loser in T. Thus, suppose that no vertex in H' has in-degree zero and hence, H' has a directed cycle. Let $L = \langle \hat{y}_{l_1}, \hat{y}_{l_2}, \ldots, \hat{y}_{l_{|L|}} \rangle$ be a directed cycle in H'. Similar to the proof of the previous condition, we leverage L to alter M to a new matching with more weight, which is a contradiction by the maximality of M.

The third condition: If $w(\hat{x}_i, \hat{y}_i) < w(\hat{x}_j, \hat{y}_i)$, we can replace the edge between $\hat{x}_i$ and $\hat{y}_i$ by $(\hat{x}_j, \hat{y}_i)$ in M which yields a matching with a greater weight. This contradicts the maximality of M. $\square$

Notice the similarities of the first and the second conditions of Lemma 2.5.14 with the conditions of the winner and loser in Definition 2.5.9. In Section 2.5.2, we assign items to the agents based on an MCMWM of the value-graph. Lemma 2.5.14 ensures that such an assignment results in a cycle-envy-free set of semi-satisfied agents.

2.5.2 Phase 1: Building the Clusters

In this section, we explain our method for clustering the agents. Intuitively, we divide the agents into three clusters $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$. As mentioned before, during the algorithm, two sets of items f_i, g_i are allocated to each agent a_i. Throughout this section, we prove a set of lemmas that are labeled as *value-lemma*. In these lemmas we bound the value of f_i and g_i allocateed to any agent for other agents. A summary of these lemmas is shown in Tables 2.2, 2.3 and 2.4.

After constructing each cluster, we refine that cluster. In the refinement phase of each cluster, we target a certain subset of the remaining items. If any item in this subset could satisfy an agent in the recently created cluster, we allocate that item to the corresponding agent. The goal of the refinement phase is to ensure that the remaining items in the targeted subset are light enough for the agents in that cluster, i.e., none of the remaining items can satisfy an agent in this cluster.

We denote by $\mathcal{S}$, the set of satisfied agents. In addition, denote by $\mathcal{S}_1, \mathcal{S}_2$, and $\mathcal{S}_3$ the subsets of $\mathcal{S}$, where $\mathcal{S}_i$ refers to the agents of $\mathcal{S}$ that previously belonged to $\mathcal{C}_i$. Furthermore, we use $\mathcal{S}_1^r$ and $\mathcal{S}_2^r$ to refer to the agents of $\mathcal{S}_1$ and $\mathcal{S}_2$ that are satisfied in the refinement phases of $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

2.5.2.1 Cluster $\mathcal{C}_1$

Consider the filtering $G_{1/2}\langle V_{1/2}(G), E_{1/2}(G)\rangle$ of the value-graph G and let M be an MCMWM of $G_{1/2}$. We define Cluster $\mathcal{C}_1$ as the set of agents whose corresponding vertex is in $N(F_{G_{1/2}}(M, \mathcal{X}_{1/2}))$.

For brevity, denote by $V_{\mathcal{C}_1}$ the set of vertices in $V(G)$ that correspond to the agents of $\mathcal{C}_1$. In other words:

$$V_{\mathcal{C}_1} = N(F_{G_{1/2}}(M, \mathcal{X}_{1/2})).$$

Also, let $F_{G_{1/2}}(M, \mathcal{X}_{1/2})$ be $U_1 \cup S_1$, where U_1 is the set of unsaturated vertices in $F_{G_{1/2}}(M, \mathcal{X}_{1/2})$ and S_1 is the set of the saturated vertices. For each edge $(x_j, y_i) \in M$ such that $x_j \in S_1$, we allocate item b_j to agent a_i. More precisely, we set $f_i = \{b_j\}$. Since $w(x_j, y_i) \geq 1/2$, we have:

$$\forall a_k \in \mathcal{C}_1 \qquad V_k(f_k) \geq 1/2.$$

According to the definition of ϵ_i, we have

$$\forall a_k \in \mathcal{C}_1 \qquad \epsilon_k \leq 1/4. \tag{2.2}$$

By the definition of $F_{G_{1/2}}$, for every agent which is not in $\mathcal{C}_1$, the condition of Lemma 2.5.15 holds. Note that all the agents that are not in $\mathcal{C}_1$, belong to either $\mathcal{C}_2$ or $\mathcal{C}_3$.

Lemma 2.5.15 (value-lemma) *For all $a_i \in \mathcal{C}_2 \cup \mathcal{C}_3$ we have:*

$$\forall a_j \in \mathcal{C}_1 \qquad V_i(f_j) < 1/2.$$

Proof. By definition, there is no edge between the vertices of $F_{G_{1/2}}(M, \mathcal{X}_{1/2})$ and $\mathcal{Y}_{1/2} \setminus N(F_{G_{1/2}}(M, \mathcal{X}_{1/2}))$ in $G_{1/2}$. Furthermore, all the items are in worth less than $1/2$ for the agents corresponding to the vertices in $\mathcal{Y} \setminus \mathcal{Y}_{1/2}$. Thus, for every agent a_i and every item b_j with $y_i \in \mathcal{Y} \setminus N(F_{G_{1/2}}(M, \mathcal{X}_{1/2}))$ and $x_j \in F_{G_{1/2}}(M, \mathcal{X}_{1/2})$, we have $V_i(b_j) < 1/2$. According to the fact that the agents that are not selected in the clustering of $\mathcal{C}_1$ either belong to $\mathcal{C}_2$ or $\mathcal{C}_3$, we have:

$$\forall a_j \in \mathcal{C}_1 \qquad V_i(f_j) < 1/2.$$

$\square$

For each vertex $y_i \in V_{\mathcal{C}_1}$, denote by N_{y_i} the set of vertices $x_j \in \mathcal{X} \setminus \mathcal{X}_{1/2}$, where $w(x_j, y_i) \geq \epsilon_i$ and let

$$W_1 = U_1 \cup \bigcup_{y_i \in V_{\mathcal{C}_1}} N_{y_i}.$$

Note that by definition, for any vertex $x_j \in U_1$ and $y_i \notin V_{\mathcal{C}_1}$, there is no edge between x_j and y_i in $G_{1/2}$ and hence $w(x_j, y_i) < 1/2$. Also, since the rest of the vertices in W_1 are from $\mathcal{X} \setminus \mathcal{X}_{1/2}$, for any vertex y_i and $x_j \in (W_1 \setminus U_1)$, $w(x_j, y_i) < 1/2$ holds. Thus, we have the following observation:

Observation 2.5.16 *For every item b_j with $x_j \in W_1$ and every agent a_i with $y_i \notin V_{\mathcal{C}_1}$,*

$V_i(\{b_j\}) < 1/2$.

Now, define $\mathcal{X}'$ and $\mathcal{Y}'$ as follows:

$$\mathcal{X}' = \mathcal{X} \setminus (W_1 \cup S_1),$$

$$\mathcal{Y}' = \mathcal{Y} \setminus V_{\mathcal{C}_1}.$$

Let $G'\langle V(G'), E(G')\rangle$ be the induced subgraph of G on $V(G') = \mathcal{Y}' \cup \mathcal{X}'$. We use graph G' to build Cluster $\mathcal{C}_2$.

2.5.2.2 Cluster $\mathcal{C}_1$ Refinement

Before building Cluster $\mathcal{C}_2$, we satisfy some of the agents in $\mathcal{C}_1$ with the items corresponding to the vertices of W_1. Consider the subgraph $G_1\langle V(G_1), E(G_1)\rangle$ of G with $V(G_1) = W_1 \cup V_{\mathcal{C}_1}$. In G_1, There is an edge between $y_i \in V_{\mathcal{C}_1}$ and $x_j \in W_1$, if $V_i(\{b_j\}) \geq \epsilon_i$. Note that $G_1\langle V(G_1), E(G_1)\rangle$ is not necessarily an induced subgraph of G. We use G_1 to satisfy a set of agents in $\mathcal{C}_1$. To this end, we first show that G_1 admits a special type of matching, described in Lemma 2.5.17.

Lemma 2.5.17 *There exists a matching M_1 in G_1, that saturates all the vertices of W_1 and for any edge $(x_i, y_j) \in M_1$ and any unsaturated vertex $y_k \in N(x_i)$, a_k does not envy a_j.*

Proof. First, we prove Lemma 2.5.18. This lemma ensures that there exists a matching in G_1 that saturates all the vertices in W_1. Lemma 2.5.18 is a consequence of irreducibility.

In fact, we show that if the condition in Lemma 2.5.18 does not hold, the instance is reducible.

Lemma 2.5.18 *For graph G_1, we have*

$$\forall R \subseteq W_1, \qquad |N(R)| > |R|.$$

Proof. Let M_1 a matching with the maximum number of edges in G_1 . Regarding Lemma 2.5.7, it only suffices to show that $F_{G_1}(M_1, W_1)$ is empty. For the sake of contradiction, suppose that $F_{G_1}(M_1, W_1)$ is not empty. As mentioned before, there exists a matching between $F_{G_1}(M_1, W_1)$ and $N(F_{G_1}(M_1, W_1))$ that saturates all the vertices in $N(F_{G_1}(M_1, W_1))$. Let

$$M_S = \{(x_{j_1}, y_{i_1}), (x_{j_2}, y_{i_2}), \ldots, (x_{j_k}, y_{i_k})\}$$

be this matching. We show that the set of agents

$$T = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$$

and the set of items

$$S = \{f_{i_1}, b_{j_1}, f_{i_2}, b_{j_2}, \ldots, f_{i_k}, b_{j_k}\}.$$

have all three conditions in Lemma 2.5.3 (Note that f_{i_l} contains exactly one item). The first condition is trivial: $|S| = 2|T|$. Regarding the definition of an edge in G_1, we know that $f_{i_l} \cup \{b_{j_l}\}$ satisfy a_{i_l} and hence, the second condition is held as well. For the third condition, we should prove that for every agent a_{i_l} in T,

$$V_{i'}(f_{i_l} \cup \{b_{j_l}\}) < 1 \qquad \forall a_{i'} \notin T.$$

To show this, we consider two cases separately. First, if $a_{i'} \notin C_1$, by Lemma 2.5.15, $V_{i'}(f_{i_l}) < 1/2$ and by Observation 2.5.16, $V_{i'}(\{b_{j_l}\}) < 1/2$, which means $V_{i'}(f_{i_l} \cup \{b_{j_l}\}) < 1$.

Moreover, consider the case that $a_{i'} \in C_1$. Note that since $a_{i'} \notin T$, it's corresponding vertex $y_{i'}$ is not in $N(F_{G_1}(M_1, W_1))$, which means:

$$y_{i'} \in V_{C_1} \setminus N(F_{G_1}(M_1, W_1)).$$

By the definition of $N(F_{G_1}(M_1, W_1)$, there is no edge between $y_{i'}$ and x_{j_l} and hence, $V_{i'}(\{b_{j_l}\}) < \epsilon_{i'} \leq 1/4$. On the other hand, by the irreducibility assumption and the fact that f_{i_l} contains exactly one item, $V_{i'}(f_{i_l}) < 3/4$. Thus, $V_{i'}(f_{i_l} \cup \{b_{j_l}\}) < 1$.

As a result, $V_{i'}(f_{i_l} \cup \{b_{j_l}\}) < 1$ for every agent $a_{i'} \notin T$ which means the third condition of Lemma 2.5.3 is held as well. Thus, regarding Lemma 2.5.3, the instance is reducible. But this contradicts the irreducibility assumption. $\square$

The rest of the proof of Lemma 2.5.17 is as follows. Since we used MCMWM to build cluster $\mathcal{C}_1$, regarding Lemma 2.5.14, $\mathcal{C}_1$ is cycle-envy-free. Consider the topological ordering of $\mathcal{C}_1$ and let p_{a_i} be the position of a_i in this ordering. More precisely, $p_{a_i} = k$ if a_i is the k-th agent in the topological ordering of $\mathcal{C}_1$.

According to Lemma 2.5.18, the condition of Hall's Theorem holds for graph G_1

and as a result there exists a matching in G_1 that saturates all the vertices in W_1. Among all possible maximum matchings of G_1, let M_1 be a maximum matching that minimizes

$$p_{M_1} = \sum_{y_i \in M_1} p_{a_i}.$$

We claim that M_1 is the desired matching described in Lemma 2.5.17. To prove our claim, we must show that for any edge $(x_i, y_j) \in M_1$ and any unsaturated vertex $y_k \in N(x_i)$, a_j is a loser for the set $\{a_j, a_k\}$, which means a_k does not envy a_j. Note that if a_k envies a_j, a_k appears before a_j in the topological ordering of $\mathcal{C}_1$ which means $p_{a_k} < p_{a_j}$. Therefore, if we replace (x_i, y_j) by (x_i, y_k) in M_1, p_{M_1} will be decreased that contradicts the minimality of p_{M_1}. $\square$

Let M_1 be a matching of G_1 with the property described in Lemma 2.5.17. For every edge $(y_i, x_j) \in M_1$, we allocate item b_j to agent a_i i.e., we set $g_i = \{b_j\}$. By the definition, a_i is now satisfied. Thus, we remove a_i from $\mathcal{C}_1$ and add it to $\mathcal{S}$. Note that, after refining $\mathcal{C}_1$, none of the items whose corresponding vertex is in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$ can satisfy any remaining agent in $\mathcal{C}_1$. Thus, Observation 2.5.19 holds.

Observation 2.5.19 *For every item b_j such that $x_j \in \mathcal{X}'$, either $x_j \in \mathcal{X}'_{1/2}$ or for all $a_i \in \mathcal{C}_1$, $V_i(\{b_j\}) < \epsilon_i$.*

At this point, all the agents of $\mathcal{S}$ belong to $\mathcal{S}_1^r$. Each one of these agents is satisfied with two items, i.e., for any agent $a_j \in \mathcal{S}_1^r$, $|f_j| = |g_j| = 1$. In Lemma 2.5.20 we give an upper bound on $V_i(g_j)$ for every agent $a_j \in \mathcal{S}_1^r$ and every agent a_i in $\mathcal{C}_2 \cup \mathcal{C}_3$.

Lemma 2.5.20 (value-lemma) *For every agent $a_i \in \mathcal{C}_2 \cup \mathcal{C}_3$, we have*

$$\forall a_j \in \mathcal{S}_1^r \qquad V_i(g_j) < 1/2.$$

Proof. Let b_k be the item assigned to a_j in the refinement of $\mathcal{C}_1$. Since $x_k \in W_1$, according to Observation 2.5.16, $V_i(g_j) < 1/2$. □

Lemmas 2.5.20 and 2.5.15 state that for every agent $a_i \in \mathcal{C}_2 \cup \mathcal{C}_3$ and every agent $a_j \in \mathcal{S}_1^r$, $V_i(f_j)$ and $V_i(g_j)$ are upper bounded by $1/2$. This, together with the fact that $|f_j| = |g_j| = 1$, results in Lemma 2.5.21.

Lemma 2.5.21 *For all $a_i \notin \mathcal{C}_1$, we have*

$$\mathsf{MMS}_{V_i}^{|\mathcal{N} \setminus \mathcal{S}_1^r|}(\mathcal{M} \setminus \bigcup_{y_j \in \mathcal{S}_1^r} f_j \cup g_j) \geq 1.$$

Proof. Let a_j be an agent in $\mathcal{S}_1^r$. First, note that $|f_j| = |g_j| = 1$. Lemma 2.5.15 together with Observation 2.5.16 state that $V_i(f_j \cup g_j) < 1$. According to Inequality (2.1), we have

$$\mathsf{MMS}_{V_i}^{|\mathcal{N} \setminus a_j|}(\mathcal{M} \setminus f_j \cup g_j) \geq 1. \tag{2.3}$$

Note that Equation (2.3) holds for every agent in $\mathcal{S}_1^r$. Applying Equation (2.3) to all the agents of $\mathcal{S}_1^r$ yields

$$\mathsf{MMS}_{V_i}^{|\mathcal{N} \setminus \mathcal{S}_1^r|}(\mathcal{M} \setminus \bigcup_{y_i \in \mathcal{S}_1^r} f_i \cup g_i) \geq 1.$$

□

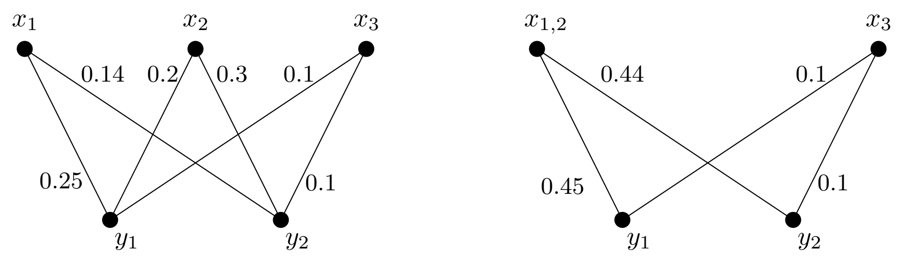


Figure 2.9: Merging x_1 and x_2

2.5.2.3 Cluster $\mathcal{C}_2$

Recall graph $G'\langle V(G'), E(G')\rangle$ as described in the last part of Section 2.5.2.1 and let $G'_{1/2}\langle V_{1/2}(G'), E_{1/2}(G')\rangle$ be a $1/2$-filtering of G'. Lemma 2.5.6 states that the size of the maximum matching between $\mathcal{X}'_{1/2}$ and $\mathcal{Y}'_{1/2}$ is $|\mathcal{X}'_{1/2}|$. Also, according to Corollary 2.5.8, for any maximum matching M' of $G'_{1/2}$, $F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})$ is empty. In what follows, we increase the size of the maximum matching in $G'_{1/2}$ by merging the vertices of $\mathcal{X}'\backslash\mathcal{X}'_{1/2}$ as described in Definition 2.5.22.

Definition 2.5.22 *For merging vertices x_i, x_j of $G'(\mathcal{X}', \mathcal{Y}')$, we create a new vertex labeled with $x_{i,j}$. Next, we add $x_{i,j}$ to $\mathcal{X}'$ and for every vertex $y_k \in \mathcal{Y}'$, we add an edge from y_k to $x_{i,j}$ with weight $w(y_k, x_i) + w(y_k, x_j)$. Finally we remove vertices x_i and x_j from $\mathcal{X}$. See Figure 2.9.*

In Lemmas 2.5.23 and 2.5.24, we give upper bounds on the value of the pair of items corresponding to a merged vertex. In Lemma 2.5.23, we show that the value of a merged vertex is less than $2\epsilon_i$ to every agent $a_i \in \mathcal{C}_1$. This fact is a consequence of Observation 2.5.19. Also, in Lemma 2.5.24, we prove that the value of the items corresponding to a merged vertex is less than $3/4$ to any agent. Lemma 2.5.24 is a direct consequence of $3/4$-irreducibility. In fact, we show that if the condition of Lemma 2.5.24 does not hold, then the problem can be reduced.

Lemma 2.5.23 *For any agent $a_k \in \mathcal{C}_1$ and any pair of vertices $x_i, x_j \in \mathcal{X}' \setminus \mathcal{X}'_{1/2}$, $V_k(\{b_i, b_j\}) < 2\epsilon_k$ holds. In particular, total value of the items that belong to a merged vertex is less than $2\epsilon_k$ for a_k.*

Proof. According to Observation 2.5.19, for any agent $a_k \in \mathcal{C}_1$ and for every $x_j \in \mathcal{X}' \setminus \mathcal{X}'_{1/2}$ we have $V_k(\{b_j\}) < \epsilon_k$. By additivity assumption, for any $a_k \in \mathcal{C}_1$ we have

$$\forall x_i, x_j \in \mathcal{X}' \setminus \mathcal{X}'_{1/2} \qquad V_k(\{b_i, b_j\}) < 2\epsilon_k.$$

□

Lemma 2.5.24 *For any pair of vertices $x_i, x_j \in \mathcal{X}' \setminus \mathcal{X}'_{1/2}$ and any vertex $y_k \in \mathcal{Y}$, we have $V_k(\{b_i, b_j\}) < 3/4$.*

Proof. Suppose for the sake of contradiction that the problem is $3/4$-irreducible, and there exists a vertex $y_k \in \mathcal{Y}$ such that $V_k(\{b_i, b_j\}) \geq 3/4$. According to Lemma 2.5.2 there exists an agent $a_{k'} \neq a_k$ such that

$$V_{k'}(\{b_i, b_j\}) \geq 1.$$

Since the valuations are additive, we know that one of the inequalities $V_{k'}(\{b_i\}) \geq 1/2$ or $V_{k'}(\{b_j\}) \geq 1/2$ are held, which is contradiction, since we know both x_i and x_j belong to $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$. □

Corollary 2.5.25 (of Lemma 2.5.24) *For any agent a_i with $y_i \in \mathcal{Y}$, there is at most one item b_j, with $x_j \in \mathcal{X}' \setminus \mathcal{X}'_{1/2}$ and $V_i(\{b_j\}) \geq 3/8$.*

Consider the vertices in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$. We call a pair (x_i, x_j) of distinct vertices in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$ *desirable* for $y_k \in \mathcal{Y}'$, if $w(y_k, x_i) + w(y_k, x_j) \geq 1/2$. With this in mind, consider the process described in Algorithm 1.

In each step of this process, we find an MCMWM M' of $G'_{1/2}$. Note that M' changes after each step of the algorithm. Next, we find a pair (x_i, x_j) of the vertices in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$ that is desirable for at least one agent in $T = \mathcal{Y}' \setminus N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$. If no such pair exists, we terminate the algorithm. Otherwise, we select an arbitrary desirable pair (x_i, x_j) and merge them to obtain a vertex $x_{i,j}$. According to the definition of T in Algorithm 1, merging a pair (x_i, x_j) results in an augmenting path in $G'_{1/2}$. Hence, the size of the maximum matching in $G'_{1/2}$ is increased by one. Note that after the termination of Algorithm 1, either $T = \varnothing$ or no pair of vertices in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$ is desirable for any vertex in T.

Lemma 2.5.26 *After running Algorithm 1, we have*

$$|F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})| = |N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))|.$$

Proof. We prove Lemma 2.5.26 in two steps. Firstly, we show that

$$|F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})| \leq |N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))|. \tag{2.4}$$

Furthermore, we prove

$$|F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})| \geq |N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))|. \tag{2.5}$$

Inequalities (2.5) and (2.5) yields

$$|F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})| = |N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))|. \tag{2.6}$$

To show Inequality (2.4)**,** argue that before Algorithm 1 starts, we have

$$F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}) = \varnothing$$

and

$$N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})) = \varnothing$$

and all the vertices in $\mathcal{X}'_{1/2}$ are saturated by M'. In each step of Algorithm 1, we add a new vertex to $\mathcal{X}'_{1/2}$, and the size of the maximum matching M' is increased by one. Therefore, after each step of Algorithm 1, all of the vertices in $\mathcal{X}'_{1/2}$ remain saturated by M'. Since $F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}) \subseteq \mathcal{X}'_{1/2}$, all the vertices of $F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})$ are also saturated by M', which means

$$|F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})| \leq |N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))|.$$

To prove Inequality (2.5), note that by definition, $F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})$ has a property

Algorithm 1: Merging vertices in G'

```
   Data: $G'(V(G'), E(G'))$
1  while True do
2      $M' = \mathsf{MCMWM}$ of $G'_{1/2}$;
3      Find $F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})$;
4      $T = \mathcal{Y}' \setminus N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$;
5      $Q$ = Set of all desirable pairs in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$ for the agents in $T$;
6      if $Q = \varnothing$ then
7          STOP;
8      else
9          Select an arbitrary pair $x_i, x_j$ from $Q$;
10         Merge($x_i, x_j$);
11     end
12 end
```

that there exists a matching from $F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})$ to $N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$ that saturates all the vertices of $N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$. Therefore, we have

$$|F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})| \geq |N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))|.$$

This completes the proof. □

We define Cluster $\mathcal{C}_2$ as the set of agents that correspond to the vertices of $N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$. Also, denote by $V_{\mathcal{C}_2}$ the vertices in $N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$. For each agent $a_i \in \mathcal{C}_2$, we allocate the item corresponding to $M'(y_i)$ (or pair of items in case $M'(y_i)$ is a merged vertex) to a_i.

Note that we put the rest of the agents in Cluster $\mathcal{C}_3$. Therefore, Lemma 2.5.27 holds for all the agents of $\mathcal{C}_3$.

Lemma 2.5.27 (value-lemma) *For all $a_i \in \mathcal{C}_3$ we have*

$$\forall a_j \in \mathcal{C}_2, V_i(f_j) < 1/2.$$

Proof. Firstly, we clarify what agents are in $\mathcal{C}_3$. Roughly speaking, the agents that are not selected for Clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ are in $\mathcal{C}_3$. Thus, the agents in $\mathcal{C}_3$ correspond to the vertices in

$$\mathcal{Y}' \setminus N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$$

$$= \big(\mathcal{Y}' \setminus \mathcal{Y}'_{1/2}\big) \cup \big(\mathcal{Y}'_{1/2} \setminus N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))\big).$$

The term $\mathcal{Y}' \setminus \mathcal{Y}'_{1/2}$ refers to the vertices that are filtered in $G'_{1/2}$ which means no edge with weight at least $1/2$ is incident to any of these vertices. On the other hand, for every agent $a_j \in \mathcal{C}_2$, f_j corresponds to a vertex in $F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})$. Hence, for every agent $a_j \in \mathcal{C}_2$ and every agent a_i with corresponding vertex in $\mathcal{Y}' \setminus \mathcal{Y}'_{1/2}$ we have $V_i(f_j) < 1/2$

Next, consider the term $\mathcal{Y}'_{1/2} \setminus N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$. By definition, the vertices of $F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})$ are only incident to the vertices of $N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$ in $G'_{1/2}$. Regarding the definition of an edge in $G'_{1/2}$, for every agent $a_j \in \mathcal{C}_2$ and agent a_i with $y_i \in \mathcal{Y}'_{1/2} \setminus N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$ we have $V_i(f_j) < 1/2$.

Therefore, for all $a_i \in \mathcal{C}_3$ we have:

$$\forall a_j \in \mathcal{C}_2 \qquad V_i(f_j) < 1/2.$$

□

2.5.2.4 Cluster $\mathcal{C}_2$ Refinement

The refinement phase of $\mathcal{C}_2$, is semantically similar to the refinement phase of $\mathcal{C}_1$. In the refinement phase of $\mathcal{C}_2$, we satisfy some of the agents of $\mathcal{C}_2$ by the items with vertices in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$. Note that none of the vertices in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$ is a merged vertex.

The refinement phase of $\mathcal{C}_2$ is presented in Algorithm 2. Let $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ be the topological ordering of the agents in $\mathcal{C}_2$ as described in Section 2.5.1.3. In Algorithm 2, We start with y_{i_1} and $W_2 = \varnothing$ and check whether there exists a vertex $x_j \in \mathcal{X}' \setminus (\mathcal{X}'_{1/2} \cup W_2)$ such that $V_{i_1}(\{b_j\}) \geq \epsilon_{i_1}$. If so, we add x_j to W_2 and satisfy a_{i_1} by allocating b_j to a_{i_1}. Next, we repeat the same process for y_{i_2} and continue on to y_{i_k}. Note that at the end of the process, W_2 refers to the vertices whose corresponding items are allocated to the agents that are satisfied during the refinement step of $\mathcal{C}_2$. For convenience, let $S_2 = F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})$ and define $\mathcal{X}''$ and $\mathcal{Y}''$ as follows:

$$\mathcal{X}'' = \mathcal{X}' \setminus (W_2 \cup S_2),$$

$$\mathcal{Y}'' = \mathcal{Y}' \setminus V_{\mathcal{C}_2}.$$

Let $G''\langle V(G''), E(G'')\rangle$ be the induced subgraph of G' on $V(G'') = \mathcal{X}'' \cup \mathcal{Y}''$. We use G'' to build Cluster $\mathcal{C}_3$.

Observation 2.5.28 *After running Algorithm 2, For every item b_j with $x_j \in \mathcal{X}'' \setminus \mathcal{X}''_{1/2}$ and every agent $a_i \in \mathcal{C}_2$, we have $V_i(\{b_j\}) < \epsilon_i$.*

In the following two lemmas, we give upper bounds on the value of g_i for every

Algorithm 2: Refinement of $\mathcal{C}_2$

```
   Data: $G'(V(G'), E(G'))$
   Data: $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ = Topological ordering of agents in $\mathcal{C}_2$
1  for $l : 1 \to k$ do
2      if $\exists x_j \in \mathcal{X}' \setminus (\mathcal{X}'_{1/2} \cup W_2)$ s.t. $V_{i_1}(\{b_j\}) \geq \epsilon_{i_l}$) then
3          $g_{i_l} = b_j$ ;
4          $W_2 = W_2 \cup x_j$;
5          $\mathcal{C}_2 = \mathcal{C}_2 \setminus a_{i_l}$;
6          $\mathcal{S} = \mathcal{S} \cup a_{i_l}$;
7      end
8  end
```

agent $a_i \in \mathcal{S}_2^r$. First, in Lemma 2.5.29, we show that for every agent $a_j \in \mathcal{C}_1$, $V_j(g_i)$ is upper bounded by ϵ_j. Furthermore, by the fact that the agents that are not selected for Clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ belong to Cluster $\mathcal{C}_3$, we show that $V_j(g_i)$ is upper bounded by $1/2$ for every agent $a_j \in \mathcal{C}_3$.

Lemma 2.5.29 *Let $a_i \in \mathcal{S}_2^r$ be an agent that is satisfied in the refinement phase of Cluster $\mathcal{C}_2$ and a_j be an agent in $\mathcal{C}_1$. Then, $V_j(g_i) < \epsilon_j$.*

Proof. Regarding Observation 2.5.19, after refinement of $\mathcal{C}_1$, all the items with vertex in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$ are in worth less than ϵ_j for every agent $a_j \in \mathcal{C}_1$. Furthermore, note that for every agent $a_i \in \mathcal{S}_2^r$, g_i is a single item with vertex in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$. Thus, $V_j(g_i) < \epsilon_j$ for every agent $a_j \in \mathcal{C}_1$. □

Lemma 2.5.30 (value-lemma) *Let $a_i \in \mathcal{S}_2^r$ be an agent that is satisfied in the refinement phase of Cluster $\mathcal{C}_2$ and a_j be an agent in $\mathcal{C}_3$. Then, $V_j(g_i) < 1/2$.*

Proof. According to Algorithm 2, for any agent $a_i \in \mathcal{S}_2^r$, the corresponding vertex of the only member of g_i is in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$. Therefore, for any agent $a_j \notin \mathcal{C}_1 \cup \mathcal{C}_2$ we have

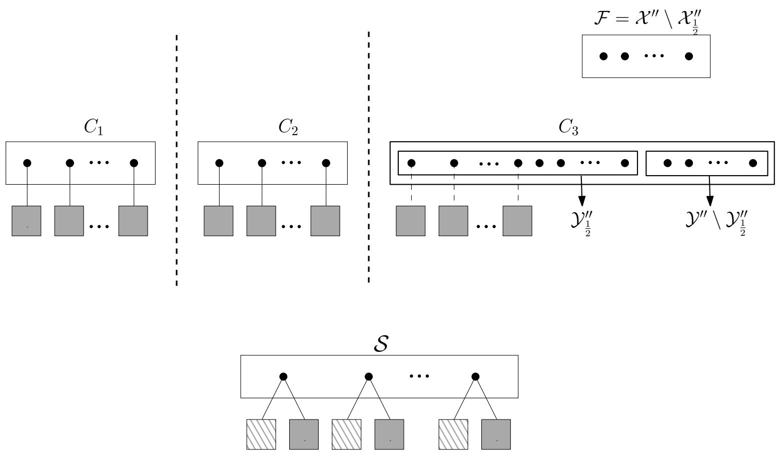
$V_j(g_i) < 1/2$. Finally, note that the remaining agents that are not in $\mathcal{C}_1$ and $\mathcal{C}_2$ belong to $\mathcal{C}_3$. $\square$

2.5.2.5 Cluster $\mathcal{C}_3$.

Finally, Cluster $\mathcal{C}_3$ is defined as the set of agents corresponding to the vertices of $\mathcal{Y}''$. Let M'' be an MCMWM of $G''_{1/2}$. Note that by Lemma 2.5.6, all the vertices in $\mathcal{X}''_{1/2}$ are saturated by M''. For each vertex y_i that is saturated by M'', we allocate the item (or pair of items in a case that $M''(y_i)$ is a merged vertex) corresponding to $M''(y_i)$ to a_i. Unlike the previous clusters, this allocation is temporary. A semi-satisfied agent a_i in $\mathcal{C}_3$ may *lend* his f_i to the other agents of $\mathcal{C}_3$. Therefore, we have three type of agents in $\mathcal{C}_3$:

1. **The semi-satisfied agents**: we denote the set of semi-satisfied agents in $\mathcal{C}_3$ by $\mathcal{C}_3^s$
2. **The borrower agents**: the agents that may borrow from a semi-satisfied agent. An agent a_j in $\mathcal{C}_3$ is a borrower, if $a_j \notin \mathcal{C}_3^s$ and $\max_{a_i \in \mathcal{C}_3^s} V_j(f_i) \geq 1/2$. We denote the set of borrower agents in $\mathcal{C}_3$ by $\mathcal{C}_3^b$.
3. **The free agents**: the remaining agents in $\mathcal{C}_3$. We denote the set of free agents by $\mathcal{C}_3^f$.

So far, the agents corresponding to unsaturated vertices in $\mathcal{Y}''_{1/2}$ belong to $\mathcal{C}_3^b$ and the agents corresponding to the vertices in $\mathcal{Y}'' \setminus \mathcal{Y}''_{1/2}$ are in $\mathcal{C}_3^f$. As we see, during the second phase, agents in $\mathcal{C}_3$ may change their type. For example, an agent in $\mathcal{C}_3^s$ may move to $\mathcal{C}_3^f$ or vice



$\mathcal{F} = \mathcal{X}'' \setminus \mathcal{X}''_{\frac{1}{2}}$
C_1
C_2
C_3
$\mathcal{Y}''_{\frac{1}{2}}$
$\mathcal{Y}'' \setminus \mathcal{Y}''_{\frac{1}{2}}$
$\mathcal{S}$

Figure 2.10: Overview on the state of the algorithm

versa. For convenience, for every agent $a_i \in \mathcal{C}_3^b$, we define ϵ_i as follows:

$$3/4 - \max_{a_j \in \mathcal{C}_3^s} V_i(f_j) \tag{2.7}$$

Note that by the definition, $\epsilon_i \leq 1/4$ holds for every agent of $\mathcal{C}_3^b$.

In Lemma 2.5.31, we show that the remaining items are not *heavy* for the agents in $\mathcal{C}_3$. The main reason that Lemma 2.5.31 holds, is the fact that no pair of vertices is desirable for any agents in $\mathcal{C}_3$ at the end of Algorithm 1.

Lemma 2.5.31 *For all $a_i \in \mathcal{C}_3$ and $x_j, x_k \in \mathcal{X}'' \setminus \mathcal{X}''_{1/2}$, we have $V_i(\{b_j, b_k\}) < 1/2$.*

Proof. The algorithm 1 terminates when there is no desirable pair for the agents in $T = \mathcal{Y}' \setminus N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2}))$. Furthermore, by definition, for every agent $a_i \in \mathcal{C}_3$ we have

$$y_i \in \mathcal{Y}' \setminus N(F_{G'_{1/2}}(M', \mathcal{X}'_{1/2})).$$

But at the end of Algorithm 1, no pair of vertices is desirable for a_i which means for every $x_j, x_k \in \mathcal{X}'' \setminus \mathcal{X}''_{1/2}$, we have $V_i(\{b_j, b_k\}) < 1/2$ (note that $\mathcal{X}'' \setminus \mathcal{X}''_{1/2} \subseteq \mathcal{X}' \setminus \mathcal{X}'_{1/2}$). $\square$

Corollary 2.5.32 (of Lemma 2.5.31) *For any agent $a_i \in \mathcal{C}_3$, there is at most one vertex $x_j \in \mathcal{X}'' \setminus \mathcal{X}''_{1/2}$, such that $V_i(\{b_j\}) \geq 1/4$.*

2.5.3 Phase 2: Satisfying the Agents

2.5.3.1 An Overview on the State of the Algorithm

Before going through the second phase, we present an overview of the current state of the agents and items. In Figure 2.10, for every agent $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{S}$, f_i is shown by a gray rectangle and for every agent $a_i \in \mathcal{S}$, g_i is shown by a hatched rectangle.

Currently, we know that every agent in $\mathcal{S}$ belongs to $\mathcal{S}_1^r$ or $\mathcal{S}_2^r$. These agents are satisfied in the refinement phases of $\mathcal{C}_1$ and $\mathcal{C}_2$. The rest of the agents will be satisfied in the second phase. For brevity, for $i \leq 2$ we use $\mathcal{S}_i^s$ to refer to the agents in $\mathcal{S}_i$ that are satisfied in the second phase. More formally,

$$\text{for } i = 1,2 \qquad \mathcal{S}_i^s = \mathcal{S}_i \setminus \mathcal{S}_i^r.$$

Since we didn't refine Cluster $\mathcal{C}_3$, all the agents in the Cluster $\mathcal{C}_3$ are satisfied in the second phase. As mentioned in the previous section, the item allocation to the semi-satisfied agents in $\mathcal{C}_3$ is temporary; That is, we may alter such allocations later. Therefore, in Figure 2.10 we illustrate such allocations by dashed lines.

In this section, we denote the set of free items (the items corresponding to the vertices in $\mathcal{X}'' \setminus \mathcal{X}''_{1/2}$ at the end of the first phase) by $\mathcal{F}$. By Observations 2.5.19, 2.5.28 and Corollary 2.5.32, we know that the items in $\mathcal{F}$ have the following properties:

	$\forall a_i \in \mathcal{C}_1$	$\forall a_i \in \mathcal{C}_2$	$\forall a_i \in \mathcal{C}_3$
$\forall a_j \in \mathcal{C}_1$	-	$V_i(f_j) < 1/2$ ($\star$)	$V_i(f_j) < 1/2$ ($\star$)
$\forall a_j \in \mathcal{C}_2$	$V_i(f_j) < 3/4$ (‡)	-	$V_i(f_j) < 1/2$ (†)
$\forall a_j \in \mathcal{C}_3^s$	$V_i(f_j) < 3/4$(‡)	$V_i(f_j) < 3/4$(‡)	-

$\star$: Lemma 2.5.15 †: Lemma 2.5.27 ‡: Lemma 2.5.33

Table 2.2: Summary of value lemmas for f_i

	$\forall a_i \in \mathcal{C}_1$	$\forall a_i \in \mathcal{C}_2$	$\forall a_i \in \mathcal{C}_3$
$\forall a_j \in \mathcal{S}_1^r$	-	$V_i(g_j) < 1/2$ ($\star$)	$V_i(g_j) < 1/2$ ($\star$)
$\forall a_j \in \mathcal{S}_2^r$	$V_i(g_j) < \epsilon_i$(†)	-	$V_i(g_j) < 1/2$ (‡)

$\star$: Lemma 2.5.20 †: Lemma 2.5.29 ‡: Lemma 2.5.30

Table 2.3: Summary of value lemmas for the agents in $\mathcal{S}_i^r$

1. For every agent a_i in $\mathcal{C}_1$, $V_i(\{b_j\}) < \epsilon_i$ holds for all $b_j \in \mathcal{F}$ (Observation 2.5.19).

2. For every agent a_i in $\mathcal{C}_2$, $V_i(\{b_j\}) < \epsilon_i$ holds for all $b_j \in \mathcal{F}$ (Observation 2.5.28).

3. For every agent a_i in $\mathcal{C}_3$, there is at most one item $b_j \in \mathcal{F}$, such that $V_i(\{b_j\}) \geq 1/4$ (Corollary 2.5.32).

In summary, items of $\mathcal{F}$ are small enough, therefore we can run a process similar to the bag filling algorithm described earlier to allocate them to the agents. Recall that our clustering and refinement methods preserve the conditions stated in Lemmas 2.5.15, 2.5.20, 2.5.27, 2.5.29 and 2.5.30. In addition to this, we state Lemma 2.5.33 as follows.

Lemma 2.5.33 (value-lemma) *For every agent $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s$, we have*

$$\forall a_j \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \qquad V_j(f_i) < 3/4.$$

Proof. At this point, for every agent $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s$, $|f_j| \leq 2$. If $|f_i| = 1$ holds, then according to Lemma 2.5.1, value of the item in f_i is less than $3/4$ to all other agents.

Moreover, if $|f_i| = 2$, then f_i corresponds to a merged vertex. In this case, by Lemmas 2.5.23 and 2.5.24, value of f_i is less than $3/4$ to all other agents. □

A brief summary of Lemmas 2.5.15, 2.5.20, 2.5.27, 2.5.29, 2.5.30 and 2.5.33 is illustrated in Tables 2.2 and 2.3. Moreover, since sets $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3^s$ are cycle-envy-free, Observation 2.5.13 holds for these sets.

2.5.3.2 Second Phase: bag filling

We begin this section with some definitions. In the following, we define the notion of feasible subsets and, based on that, we define $\phi(S)$ for a feasible subset S of items.

Definition 2.5.34 *A subset S of items in $\mathcal{F}$ is feasible, if at least one of the following conditions are met:*

- *There exists an agent $a_i \in \mathcal{C}_3^f$ such that $V_i(\{S\}) \geq 1/2$.*
- *There exists an agent $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s \cup \mathcal{C}_3^b$ such that $V_i(\{S\}) \geq \epsilon_i$.*

Definition 2.5.35 *For a feasible set S, we define $\Phi(S)$ as the set of agents, that set S is feasible for them.*

Recall the notion of cycle-envy-freeness and the topological ordering of the agents in a cycle-envy-free set of semi-satisfied agents. Based on this, we define a total order $\prec_{pr}$ to prioritize the agents in the bag filling algorithm.

Definition 2.5.36 *Define a total order $\prec_{pr}$ on the agents of $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3$ with the following rules:*

- $a_{i_5} \prec_{pr} a_{i_1} \prec_{pr} a_{i_2} \prec_{pr} a_{i_3} \prec_{pr} a_{i_4} \qquad \forall a_{i_1} \in \mathcal{C}_1, a_{i_2} \in \mathcal{C}_2, a_{i_3} \in \mathcal{C}_3^s, a_{i_4} \in \mathcal{C}_3^b, a_{i_5} \in \mathcal{C}_3^f$

- $a_i \prec_{pr} a_j \Leftrightarrow a_i \prec_o a_j \qquad \forall a_i, a_j \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s, a_i, a_j$ *in the same cluster*

- $a_i \prec_{pr} a_j \Leftrightarrow i < j \qquad \forall a_i, a_j \in \mathcal{C}_3^b \vee a_i, a_j \in \mathcal{C}_3^f$

Recall that $\prec_o$ refers to the topological ordering of a semi-satisfied set of agents. Roughly speaking, for the semi-satisfied agents in the same cluster, $\prec_{pr}$ behaves in the same way as $\prec_o$. Furthermore, for the agents in different clusters, agents in $\mathcal{C}_3^f, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3^s, \mathcal{C}_3^b$ have a lower priority, respectively. Finally, the order of the agents in $\mathcal{C}_3^b$ and $\mathcal{C}_3^f$ is determined by their index, i.e., the agent with a lower index has a lower priority.

The second phase consists of several rounds and every round has two steps. Each of these two steps is described below. We continue running this algorithm until $\mathcal{F}$ is no longer feasible for any agent.

- **Step1**: In the first step, we run a process very similar to the bag filling algorithm described in Section 2.1. That is, we find a feasible subset $S \subseteq \mathcal{F}$, such that $|S|$ is minimal. Such a subset can easily be found, using a slight modification of the bag filling process (see Section 2.5.5.2).

- **Step2**: In the second step, we choose an agent to allocate set S to him. In contrast to the bag filling algorithm, we do not select an arbitrary agent. Instead, we select the agent in $\Phi(S)$ with the lowest priority regarding $\prec_{pr}$, i.e., smallest element in $\Phi(S)$ regarding $\prec_{pr}$. Let a_i be the selected agent. We consider three cases separately:

  - $a_i \in \mathcal{C}_3^f$: temporarily allocate S to a_i, i.e., set $f_i = S$.

- $a_i \in \mathcal{C}_3^b$: let a_j be the agent that $V_i(f_j) = 3/4 - \epsilon_j$. Take back f_j from a_j and allocate $f_j \cup S$ to a_i i.e. set $f_i = f_j$, $f_j = \varnothing$ and $g_i = S$. Remove a_i from $\mathcal{C}_3$ and add it to $\mathcal{S}$.
- $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s$: satisfy agent a_i by S, i.e., set $g_i = S$ and remove a_i from its corresponding cluster and add it to $\mathcal{S}$.

By the construction of $\mathcal{C}_3^s, \mathcal{C}_3^b$, and $\mathcal{C}_3^f$, the above process may cause agents in $\mathcal{C}_3$ to move in between $\mathcal{C}_3^s, \mathcal{C}_3^b$ and $\mathcal{C}_3^f$. For example, if the first case happens, then a_i is moved from $\mathcal{C}_3^f$ to $\mathcal{C}_3^s$. In addition, all other agents in $\mathcal{C}_3^f$ for which S is feasible are moved to $\mathcal{C}_3^b$. For the second case, a_j is moved to one of $\mathcal{C}_3^f$ or $\mathcal{C}_3^b$, based on $V_j(f_k)$ for every $a_k \in \mathcal{C}_3^s$; that is, if there exists an agent $a_k \in \mathcal{C}_3^s$ such that $V_j(f_k) \geq 1/2$, a_j is moved to $\mathcal{C}_3^b$. Otherwise, a_j is moved to $\mathcal{C}_3^f$. For both the second and the third cases, some of the agents in $\mathcal{C}_3^b$ may move to $\mathcal{C}_3^f$.

The second phase terminates, when $\mathcal{F}$ is no longer feasible for any agent. More details about the second phase can be found in Algorithm 3. In Algorithm 3, we use $Update(\mathcal{C}_3)$ to refer the process of moving agents among $\mathcal{C}_3^s, \mathcal{C}_3^b$ and $\mathcal{C}_3^f$.

In each round of the second phase, either an agent is satisfied or an agent in $\mathcal{C}_3^f$ becomes semi-satisfied. In Lemma 2.5.37, we show that if an agent $a_i \in \mathcal{C}_3^f$ is selected in some round of the second phase, then $V_j(f_i)$ is upper bounded by $2\epsilon_j$ for every agent $a_j \in \mathcal{C}_3 \cup \mathcal{C}_2 \cup \mathcal{C}_1^s \cup \mathcal{C}_1^b$. As a consequence of Lemma 2.5.37, in Lemma 2.5.38 we show that sets $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ remain cycle-envy-free during the second phase. For convenience, we use $\mathbb{R}_z$ to refer to the z'th round of the second phase.

Algorithm 3: The Second Phase

```
Data: $\mathcal{F}, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$
while $\mathcal{F}$ is feasible do
    $S$ = a minimal feasible subset of $\mathcal{F}$ ;
    $a_i =$ agent in $\Phi(S)$ with lowest order regarding $\prec_{pr}$;
    if $a_i \in C_3^f$ then
        $f_i = S$ ;
        $Update(\mathcal{C}_3)$ ;
    end
    if $a_i \in \mathcal{C}_3^b$ then
        Let $a_j$ be the agent that $V_i(f_j) = 3/4 - \epsilon_i$ ;
        $f_i = f_j$ ;
        $g_i = S$ ;
        $\mathcal{S} = \mathcal{S} \cup a_i$ ;
        $f_j = \varnothing$;
        $\mathcal{C}_3 = \mathcal{C}_3 \setminus a_i$ ;
        $Update(\mathcal{C}_3)$ ;
    end
    if $a_i \in \mathcal{C}_3^s$ then
        $g_i = S$;
        $\mathcal{S} = \mathcal{S} \cup a_i$;
        $\mathcal{C}_3 = \mathcal{C}_3 \setminus a_i$ ;
        $Update(\mathcal{C}_3)$ ;
    end
    if $a_i \in \mathcal{C}_1 \cup \mathcal{C}_2$ then
        $g_i = S$;
        remove $a_i$ from its corresponding cluster ;
        $\mathcal{S} = \mathcal{S} \cup a_i$;
    end
end
```

Lemma 2.5.37 *Let $\mathbb{R}_z$ be a round of the second phase that an agent $a_i \in \mathcal{C}_3^f$ is selected. Then, for every agent $a_j \in \mathcal{C}_3 \cup \mathcal{C}_2 \cup \mathcal{C}_1^s \cup \mathcal{C}_1^b$, we have $V_j(f_i) < 2\epsilon_j < 3/4$.*

Proof. According to Lemma 2.5.31, value of every pair of items in $\mathcal{F}$ is less than $1/2$ to a_i. Therefore, f_i contains at least three items. Let b_k be an arbitrary item in f_i. Since $|f_i| \geq 3$, $f_i \setminus \{b_k\}$ is non-empty. On the other hand, S is minimal and hence, none of the sets $f_i \setminus b_k$ and $\{b_k\}$ is feasible for any agent. According to the definition of feasibility for the agents of $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s \cup \mathcal{C}_3^b$, we have

$$\forall a_j \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s \cup \mathcal{C}_3^b \qquad V_j(f_i \setminus \{b_k\}) < \epsilon_j$$

and

$$\forall a_j \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s \cup \mathcal{C}_3^b \qquad V_j(\{b_k\}) < \epsilon_j$$

which means

$$\forall a_j \in \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s \cup \mathcal{C}_3^b \qquad V_j(f_i) < 2\epsilon_j.$$

$\square$

Lemma 2.5.38 *During the second phase, the $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3^s$ maintain the property of cycle-envy-freeness.*

Proof. The Lemma trivially holds for $\mathcal{C}_1$ and $\mathcal{C}_2$, since removing an agent from a cycle-envy-free set preserves this property. For $\mathcal{C}_3^s$, there may be multiple rounds that an agent is added to $\mathcal{C}_3^s$. We show that adding an agent to $\mathcal{C}_3^s$ preserves cycle-envy-freeness as well.

For the sake of contradiction, let $\mathbb{R}_z$ be the first round in which adding an agent a_i to $\mathcal{C}_3^s$ results in a set, that is no longer cycle-envy-free. Since $\mathcal{C}_3^s \setminus \{a_i\}$ is cycle-envy-free, every subset of $\mathcal{C}_3^s \setminus \{a_i\}$ contains at least one winner and one loser. Moreover, by Lemma 2.5.37 we have:

$$\forall a_j \in \mathcal{C}_3^s, j \neq i, \qquad V_j(f_i) < 2\epsilon_j. \tag{2.8}$$

Note that a_i previously belonged to $\mathcal{C}_3^f$. By definition of $\mathcal{C}_3^f$

$$\forall a_j \in \mathcal{C}_3^s, j \neq i, \qquad V_i(f_j) < 1/2. \tag{2.9}$$

Inequalities (2.8) and (2.9) together imply that a_i is both a winner and a loser for every subset of $\mathcal{C}_3^s$ that contains a_i. This means that every subset of $\mathcal{C}_3^s$ contains at least one winner and one loser, which is a contradiction.

□

Finally, for the rounds that an agents a_i is satisfied, Lemmas 2.5.39 and 2.5.40 give upper bounds on the value of g_i for remaining agents in different clusters.

Lemma 2.5.39 (value-lemma) *Let $a_i \in \mathcal{S}$ be an agent that is satisfied in the second phase. Then, for every other agent $a_j \in \mathcal{C}_1 \cup \mathcal{C}_2$ we have:*

- *If $a_j \prec_{pr} a_i$, then $V_j(g_i) < \epsilon_j$.*
- *If $a_i \prec_{pr} a_j$, then $V_j(g_i) < 2\epsilon_j$.*

Proof.

If $a_j \prec_{pr} a_i$, then g_i is not feasible for a_j, since the agent with the lowest priority is satisfied in each round of the second phase. Thus, $V_j(g_i) < \epsilon_j$. For the case where $a_i \prec_{pr} a_j$, let b_k be an arbitrary item of g_i. According to the fact that g_i is minimal, $g_i \setminus \{b_k\}$ is not feasible for any agent. Hence, $V_j(g_i \setminus \{b_k\}) < \epsilon_j$. On the other hand, by Observations 2.5.19 and 2.5.28, $V_j(\{b_k\}) < \epsilon_j$. Therefore, $V_j(g_i) < 2\epsilon_j$. □

	$\forall a_i \in \mathcal{C}_1$	$\forall a_i \in \mathcal{C}_2$	$\forall a_i \in \mathcal{C}_3$
$\forall a_j \in \mathcal{S}_1^s$	-	$V_i(g_j) < 2\epsilon_i(\star)$	$V_i(g_j) < 1/2(\dagger)$
$\forall a_j \in \mathcal{S}_2^s$	$V_i(g_j) < \epsilon_i(\star)$	-	$V_i(g_j) < 1/2(\dagger)$
$\forall a_j \in \mathcal{S}_3$	$V_i(g_j) < \epsilon_i(\star)$	$V_i(g_j) < \epsilon_i(\star)$	-

$\star$: Lemma 2.5.39 $\quad\dagger$: Lemma 2.5.40

Table 2.4: Summary of value lemmas for g_i

Lemma 2.5.40 (value-lemma) *Let a_i be an agent in $\mathcal{S}_1^s \cup \mathcal{S}_2^s$. Then, for every agent $a_j \in \mathcal{C}_3$, we have $V_j(g_i) < 1/2$.*

Proof. Let $\mathbb{R}_z$ be the round, in which a_i is satisfied. At that point, if $a_j \in \mathcal{C}_3^f$ then $V_j(g_i) < 1/2$ trivially holds. Since in round $\mathbb{R}_z$, $a_j \prec_{pr} a_i$ holds, g_i was not feasible for a_j in the first place. Recall that in each round, the agent with lowest order in $\Phi(S)$ is selected.

Furthermore, if in round $\mathbb{R}_z$, a_j was in $\mathcal{C}_3^s \cup \mathcal{C}_3^b$, according to Observations 2.5.19 and 2.5.28, $|S| \geq 2$, since no item alone can satisfy a_i. If $|S| = 2$, then by Observation 2.5.31, $V_j(g_i) < 1/2$. For the case of $|S| > 2$, let b_k be the item in S with the minimum value to a_j. According to Corollary 2.5.32, $V_j(\{b_k\}) < 1/4$. Also, since S is minimal, $S \setminus \{b_k\}$ is not feasible for any agent and hence, $V_j(S \setminus \{b_k\}) < \epsilon_j \leq 1/4$. Thus, $V_j(S) < 1/2$. $\square$

The results of Lemmas 2.5.39 and 2.5.40 are summarized in Table 2.4.

2.5.4 The Algorithm Finds a $3/4$-MMS Allocation

In the rest of this section, we prove that the algorithm finds a $3/4$-MMS allocation. For the sake of contradiction, suppose that the second phase is terminated, which means $\mathcal{F}$ is not feasible anymore, but not all agents are satisfied. Such an unsatisfied agent belongs to one of the Clusters $\mathcal{C}_1$ or $\mathcal{C}_2$, or $\mathcal{C}_3$. In Lemmas 2.5.41, 2.5.45, and 2.5.47, we separately rule out each of these possibilities. This implies that all the agents are satisfied and contradicts the assumption. We begin with Cluster $\mathcal{C}_3$.

Lemma 2.5.41 *At the end of the algorithm we have $\mathcal{C}_3 = \varnothing$.*

To prove Lemma 2.5.41 we consider two cases separately. If $\mathcal{C}_3 \neq \varnothing$, either there exists an agent $a_i \in \mathcal{C}_3^s \cup \mathcal{C}_3^b$ or all the agents of $\mathcal{C}_3$ are in $\mathcal{C}_3^f$. If the former holds, we show $\mathcal{C}_3^s$ is non-empty and assume a_i is a winner of $\mathcal{C}_3^s$. We bound the total value of a_i for all the items dedicated to other agents and show the value of the remaining items in $\mathcal{F}$ is at least ϵ_i for a_i. This shows set $\mathcal{F}$ is feasible for a_i and contradicts the termination of the algorithm. In case all agents of $\mathcal{C}_3$ are in $\mathcal{C}_3^f$, let a_i be an arbitrary agent of $\mathcal{C}_3^f$. With a similar argument we show that the value of a_i for the remaining unassigned items is at least $3/4$ and conclude that $\mathcal{F}$ is feasible for a_i which again contradicts the termination of the algorithm.

Before proceeding to the proof of Lemma 2.5.41, we show Lemmas (2.5.42, 2.5.43 and 2.5.44).

Lemma 2.5.42 *Let a_i be an agent in $\mathcal{S}_3$ and let $\mathbb{R}_z$ be the round of the second phase in which a_i is satisfied. Then, for any other agent a_j that is in $\mathcal{C}_3^f$ in $\mathbb{R}_z$, $V_j(g_i) < 1/2$ holds.*

Proof. In $\mathbb{R}_z$, a_i either belongs to $\mathcal{C}_3^s$ or $\mathcal{C}_3^b$. Thus, $a_j \prec_{pr} a_i$, and thus g_i is not feasible for a_j in that round. Therefore, $V_j(g_i) < 1/2$. □

Lemma 2.5.43 *Let $a_i \in \mathcal{S}_3$ be a satisfied agent and let $\mathbb{R}_z$ be the round in which a_i is satisfied. Then, for every other agent a_j that belongs to $\mathcal{C}_3^s \cup \mathcal{C}_3^b$ in that round, either $V_j(g_i) < \epsilon_j$ or $V_j(f_i) \leq 3/4 - \epsilon_j$.*

Proof. If g_i is not feasible for a_j, then the condition trivially holds. Moreover, by the definition, the statement is correct for the agents of $\mathcal{C}_3^b$. Therefore, it only suffices to consider the case that $a_j \in \mathcal{C}_3^s$ and g_i is feasible for a_j. Due to the priority rules for satisfying the agents in the second phase, $a_i \prec_{pr} a_j$ and hence, a_i cannot be in $\mathcal{C}_3^b$. Thus, $a_i \in \mathcal{C}_3^s$. According to Observation 2.5.13 and the fact that $\prec_{pr}$ is equivalent to $\prec_o$ for the agents in $\mathcal{C}_3^s$, we have $V_j(f_i) \leq 3/4 - \epsilon_j$. □

Lemma 2.5.44 *During the second phase, for any agent a_i in $\mathcal{C}_3$, we have:*

$$\sum_{a_j \in \mathcal{S}_3} V_i(f_j \cup g_j) < |\mathcal{S}_3| + 1/4.$$

Proof. To show Lemma 2.5.44, we show that for all the agents $a_j \in \mathcal{S}_3$ except at most one agent, $V_i(f_j \cup g_j) < 1$ holds. To show this, let $\mathbb{R}_z$ be an arbitrary round of the second phase, in which an agent $a_j \in \mathcal{C}_3$ is satisfied. First, note that in $\mathbb{R}_z$, a_j belongs to $\mathcal{C}_3^s \cup \mathcal{C}_3^b$. Also, in round $\mathbb{R}_z$, a_i belongs to one of $\mathcal{C}_3^s, \mathcal{C}_3^b$, or $\mathcal{C}_3^f$.

If $a_i \in \mathcal{C}_3^f$, then by Lemma 2.5.42, $V_i(g_j) < 1/2$ holds. On the other hand, by definition, $V_i(f_j) < 1/2$ and hence, $V_i(f_j \cup g_j) < 1$.

Now, consider the case, where $a_i \in \mathcal{C}_3^b \cup \mathcal{C}_3^s$. Note that by Lemma 2.5.43, either $V_i(f_j) \leq 3/4 - \epsilon_i$ or $V_i(g_j) < \epsilon_i$. If $V_i(g_j) < \epsilon_i$, then by Lemmas 2.5.33 and 2.5.37, we know $V_i(f_j) < 3/4$ and hence, $V_i(f_j \cup g_j) < 3/4 + \epsilon_i < 1$.

For the case where $V_i(f_j) \leq 3/4 - \epsilon_i$, let b_l be the item in g_j with the maximum value to a_i. By minimality of g_j, $g_j \setminus \{b_l\}$ is not feasible for any agent, including a_i and thus, $V_i(g_j \setminus \{b_l\}) < \epsilon_i$. Recall that by Corollary 2.5.32, there is at most one item b_k in $\mathcal{F}$, such that $V_i(b_k) \geq 1/4$. In addition to this, $V_i(b_k) < 1/2$ trivially holds, since b_k is not assigned to any agent during the clustering phase. If $b_l \neq b_k$, $V_i(g_j) < 1/4 + \epsilon_i$ holds and hence,

$$V_i(f_j \cup g_j) < 3/4 - \epsilon_i + 1/4 + \epsilon_i < 1.$$

Moreover, If $b_l = b_k$, $V_i(g_j) < 1/2 + \epsilon_i$ holds and thus, $V_i(f_j \cup g_j) < 3/4 - \epsilon_i + 1/2 + \epsilon_i < 5/4$. But, this can happen at most one round. Therefore, for all the agents $a_j \in \mathcal{S}_3$ except at most one, $V_i(f_j \cup g_j) < 1$. Also, for at most one agent $a_j \in \mathcal{S}_3$, $V_i(f_j \cup g_j) < 5/4$. Thus,

$$\sum_{a_j \in \mathcal{S}_3} V_i(f_j \cup g_j) < |\mathcal{S}_3| + 1/4.$$

$\square$

Proof of Lemma 2.5.41: Suppose for the sake of contradiction that $\mathcal{C}_3 \neq \varnothing$. Note that, by the definition of $\mathcal{C}_3^b$, if $\mathcal{C}_3^s = \varnothing$ holds, then consequently $\mathcal{C}_3^b = \varnothing$. Therefore, since we have $\mathcal{C}_3 = \mathcal{C}_3^s \cup \mathcal{C}_3^b \cup \mathcal{C}_3^f$, if $\mathcal{C}_3$ is non-empty, at least either of the two sets $\mathcal{C}_3^s$ or $\mathcal{C}_3^f$ is non-empty. In case $\mathcal{C}_3^s$ is non-empty, let a_i be a winner of $\mathcal{C}_3^s$, otherwise let a_i be an arbitrary agent of $\mathcal{C}_3^f$.

According to Lemma 2.5.40, for every agent $a_j \in \mathcal{S}_1^s \cup \mathcal{S}_2^s$, $V_i(g_j) < 1/2$ holds. Also, by Lemmas 2.5.20 and 2.5.30, for every agent $a_j \in \mathcal{S}_1^r \cup \mathcal{S}_2^r$, we have $V_i(g_j) < 1/2$. Therefore,

$$\forall a_j \in \mathcal{S}_1 \cup \mathcal{S}_2 \qquad V_i(g_j) < 1/2.$$

Also, by Lemmas 2.5.15 and 2.5.27 we know that $V_i(f_j) < 1/2$ for every $a_j \in$

$\mathcal{S}_1 \cup \mathcal{S}_2$. Thus, for every satisfied agent $a_j \in \mathcal{S}_1 \cup \mathcal{S}_2$, $V_i(f_j \cup g_j) < 1$ holds, and hence

$$\sum_{a_j \in \mathcal{S}_1 \cup \mathcal{S}_2} V_i(f_j \cup g_j) < |\mathcal{S}_1 \cup \mathcal{S}_2|. \tag{2.10}$$

Moreover, by Lemma 2.5.44, the total value of items assigned to the agents in $\mathcal{S}_3$ to a_i is less than $|\mathcal{S}_3| + 1/4$. More precisely,

$$\sum_{a_j \in \mathcal{S}_3} V_i(f_j \cup g_j) \leq |\mathcal{S}_3| + 1/4. \tag{2.11}$$

Inequality (5.5) along with Inequality (5.6) implies:

$$\begin{aligned} \sum_{a_j \in \mathcal{S}} V_i(f_j \cup g_j) &= \sum_{a_j \in \mathcal{S}_1 \cup \mathcal{S}_2} V_i(f_j \cup g_j) + \sum_{a_j \in \mathcal{S}_3} V_i(f_j \cup g_j) \\ &< |\mathcal{S}_1 \cup \mathcal{S}_2| + |\mathcal{S}_3| + 1/4 \\ &= |\mathcal{S}| + 1/4 \end{aligned} \tag{2.12}$$

Recall that the total sum of the item values for a_i is equal to n. In addition to this, since every agent belongs to either of the Clusters $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, or $\mathcal{S}$ we have

$$|\mathcal{S}| + |\mathcal{C}_1| + |\mathcal{C}_2| + |\mathcal{C}_3| = n.$$

Furthermore, every item $b_j \in \mathcal{M}$ either belongs to $\mathcal{F}$ or one of the sets $f_{j'}$ and $g_{j'}$ for an agent $a_{j'}$. More precisely,

$$\mathcal{F} = \mathcal{M} \setminus \Big[\bigcup_{a_j \in \mathcal{S} \cup \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3^s} f_j \cup \bigcup_{a_j \in \mathcal{S}} g_j \Big].$$

Therefore

$$\begin{aligned}\sum_{a_j\in\mathcal{C}_1} V_i(f_j)+\sum_{a_j\in\mathcal{C}_2} V_i(f_j)+\sum_{a_j\in\mathcal{C}_3^s} V_i(f_j)+V_i(\mathcal{F}) &= V_i(\mathcal{M})-\sum_{a_j\in\mathcal{S}} V_i(f_j\cup g_j)\\ &= n-\sum_{a_j\in\mathcal{S}} V_i(f_j\cup g_j)\\ &\geq n-(|\mathcal{S}|+1/4)\\ &= |\mathcal{C}_1|+|\mathcal{C}_2|+|\mathcal{C}_3|-1/4\end{aligned} \tag{2.13}$$

According to Lemmas 2.5.15 and 2.5.21,

$$\sum_{a_j\in\mathcal{C}_1} V_i(f_j) < 1/2|\mathcal{C}_1| \tag{2.14}$$

and

$$\sum_{a_j\in\mathcal{C}_2} V_i(f_j) < 1/2|\mathcal{C}_2| \tag{2.15}$$

hold. Inequalities (2.13), (2.14), and (2.15) together prove

$$\begin{aligned}V_i(\mathcal{F}) &\geq |\mathcal{C}_1|+|\mathcal{C}_2|+|\mathcal{C}_3|-1/4-\big[\sum_{a_j\in\mathcal{C}_1} V_i(f_j)+\sum_{a_j\in\mathcal{C}_2} V_i(f_j)+\sum_{a_j\in\mathcal{C}_3^s} V_i(f_j)\big]\\ &\geq |\mathcal{C}_1|+|\mathcal{C}_2|+|\mathcal{C}_3|-1/4-\big[1/2|\mathcal{C}_1|+1/2|\mathcal{C}_2|+\sum_{a_j\in\mathcal{C}_3^s} V_i(f_j)\big]\\ &\geq 1/2|\mathcal{C}_1|+1/2|\mathcal{C}_2|+|\mathcal{C}_3|-1/4-\sum_{a_j\in\mathcal{C}_3^s} V_i(f_j).\end{aligned} \tag{2.16}$$

Now, we consider two cases separately: (i) $a_i\in\mathcal{C}_3^s$ and (ii) $a_i\in\mathcal{C}_3^f$.

In case $a_i \in \mathcal{C}_3^s$, since a_i is a winner of $\mathcal{C}_3^s$, we have

$$\begin{aligned}\sum_{a_j \in \mathcal{C}_3^s} V_i(f_j) &\leq \sum_{a_j \in \mathcal{C}_3^s} V_i(f_i) \\ &= \sum_{a_j \in \mathcal{C}_3^s} 3/4 - \epsilon_i \\ &= (3/4 - \epsilon_i)|\mathcal{C}_3^s|.\end{aligned} \tag{2.17}$$

This combined with Inequality (2.16) concludes

$$\begin{aligned}V(\mathcal{F}) &\geq 1/2|\mathcal{C}_1| + 1/2|\mathcal{C}_2| + |\mathcal{C}_3| - 1/4 - \sum_{a_j \in \mathcal{C}_3^s} V_i(f_j) \\ &\geq 1/2|\mathcal{C}_1| + 1/2|\mathcal{C}_2| + |\mathcal{C}_3| - 1/4 - (3/4 - \epsilon_i)|\mathcal{C}_3^s| \\ &\geq 1/2|\mathcal{C}_1| + 1/2|\mathcal{C}_2| + (1/4 + \epsilon)|\mathcal{C}_3| - 1/4.\end{aligned}$$

On the other hand, since $a_i \in \mathcal{C}_3^s$, $|\mathcal{C}_3| \geq 1$ and hence, $V_i(\mathcal{F}) \geq 1/4 + \epsilon_j - 1/4 = \epsilon_j$. This means that $\mathcal{F}$ is feasible for a_i, which contradicts the termination of the algorithm.

In case $a_i \in \mathcal{C}_3^f$, by the definition of $\mathcal{C}_3^f$ we know that $\sum_{a_j \in \mathcal{C}_3^s} V_i(f_j) < 1/2|\mathcal{C}_3^s|$, which by Inequality (2.16) implies:

$$V_i(\mathcal{F}) > 1/2|\mathcal{C}_3^s| + |\mathcal{C}_3^b| + |\mathcal{C}_3^f| + 1/2|\mathcal{C}_2| + 1/2|\mathcal{C}_1| - 1/4.$$

Since $a_i \in \mathcal{C}_3^f$, we have $|\mathcal{C}_3^f| \geq 1$ and hence, $V_i(\mathcal{F}) > 3/4$. Again, this contradicts the termination of the algorithm since $\mathcal{F}$ is feasible for a_i. □

Next, we prove a similar statement for $\mathcal{C}_1$.

Lemma 2.5.45 *At the end of the algorithm we have* $\mathcal{C}_1 = \varnothing$.

Proof of Lemma 2.5.45 follows from a coloring argument. Let a_i be a winner of $\mathcal{C}_1$. We color all items in either blue or white. Roughly speaking, blue items are in a sense *heavy*, i.e., they may have a high valuation to a_i whereas white items are somewhat *lighter* and have a low valuation to a_i. Next, via a double counting argument, we show that a_i's value for the items of $\mathcal{F}$ is at least ϵ_i and thus $\mathcal{F}$ is feasible for a_i. This contradicts $\mathcal{C}_1 = \varnothing$ and shows at the end of the algorithm all agents of $\mathcal{C}_1$ are satisfied.

Proof of Lemma 2.5.45: By Lemma 2.5.41, we already know $\mathcal{C}_3 = \varnothing$. Now, let a_i be a winner of the remaining agents in $\mathcal{C}_1$. For convenience, we color the items in either blue or white. Intuitively, blue items may have a high value for a_i whereas white items are always of lower value to a_i. Initially, all items are colored in white. For each $a_j \in \mathcal{N}$, if $|f_j| = 1$, then we color the item in f_j in blue. Moreover, for every $a_j \in \mathcal{S}$, if $|g_j| = 1$ and $V_i(g_j) \geq \epsilon_i$, then we color the item in g_j in blue.

Now, let $\mathcal{P} = \langle P_1, P_2, \dots, P_n \rangle$ be the optimal n-partitioning of the items in $\mathcal{M}$ for a_i, that is, the value of every partition P_k to a_i is at least 1. Based on the coloring procedure, we have three types of partitions in $\mathcal{P}$:

- B_2: the set of partitions with at least two blue items
- B_1: the set of partitions with exactly one blue item
- B_0: the set of partitions without any blue items

Note that every partition in $\mathcal{P}$ belongs to one of B_0, B_1 or B_2. Hence,

$$|B_0| + |B_1| + |B_2| = n \tag{2.18}$$

As declared, all the items in the partitions of B_0 are white. The total value of these items to a_i is at least $|B_0| \geq 4\epsilon_i|B_0|$, which is

$$\sum_{P_k \in B_0} \sum_{b_j \in P_k} V_i(b_j) \geq 4\epsilon_i|B_0|. \tag{2.19}$$

Also, each partition in B_2 has at least two blue items, each of which is singly assigned to another agent. We decompose the partitions of B_1 into two disjoint sets, namely $\hat{B_1}$ and $\tilde{B_1}$. More precisely, let $\hat{B_1}$ be the partitions in B_1, in which the blue item is worth more than $V_i(f_i)$ to a_i and $\tilde{B_1} = B_1 \setminus \hat{B_1}$. As such, for each partition $P_k \in \tilde{B_1}$, the white items in P_k are worth at least

$$\begin{aligned} 1 - V_i(f_i) &= 1 - (3/4 - \epsilon_i) \\ &= 1/4 + \epsilon_i \\ &\geq 2\epsilon_i \end{aligned}$$

to a_i. Therefore,

$$\sum_{P_k \in \tilde{B_1}} V_i(\mathcal{W}(P_k)) \geq 2|\tilde{B_1}|\epsilon_i \tag{2.20}$$

where $\mathcal{W}(S)$ stands for the set of white items in a set S of items. On the other hand, since the problem is $3/4$-irreducible, by Lemma 2.5.1, no item alone is of worth $3/4$ to a_i and thus for each partition $P_k \in \hat{B_1}$, the white items in P_k have a value of at least $1/4 \geq \epsilon_i$ to a_i. This implies that

$$\sum_{P_k \in \hat{B_1}} V_i(\mathcal{W}(P_k)) \geq |\hat{B_1}|\epsilon_i. \tag{2.21}$$

By Inequalities (2.18),(2.19), (2.20), and (2.21) we have

$$\begin{aligned}
V_i(\mathcal{W}(\mathcal{M})) &= \sum_{P_j \in B_0} V_i(\mathcal{W}(P_j)) + \sum_{P_j \in B_1} V_i(\mathcal{W}(P_j)) + \sum_{P_j \in B_2} V_i(\mathcal{W}(P_j)) \\
&\geq \sum_{P_j \in B_0} V_i(\mathcal{W}(P_j)) + \sum_{P_j \in B_1} V_i(\mathcal{W}(P_j)) \\
&\geq \sum_{P_j \in B_0} V_i(\mathcal{W}(P_j)) + \sum_{P_j \in \hat{B}_1} V_i(\mathcal{W}(P_j)) + \sum_{P_j \in \tilde{B}_1} V_i(\mathcal{W}(P_j)) \\
&\geq |B_0| 4\epsilon_i + |\hat{B}_1|\epsilon_i + |\tilde{B}_1| 2\epsilon_i \\
&\geq |B_0| 4\epsilon_i + |B_1| 2\epsilon_i - |\hat{B}_1|\epsilon_i \\
&\geq |B_0| 4\epsilon_i + |B_1| 4\epsilon_i + |B_2| 4\epsilon_i - |B_1| 2\epsilon_i - |B_2| 4\epsilon_i - |\hat{B}_1|\epsilon_i \\
&= (2n - 2|B_2| - |B_1| - |\hat{B}_1|) 2\epsilon_i + (|\hat{B}_1|)\epsilon_i
\end{aligned} \tag{2.22}$$

Note that the total value of white items that are assigned to the agents during the algorithm is equal to $V_i(\mathcal{W}(\mathcal{M} \setminus \mathcal{F}))$. The rest of the white items are still in $\mathcal{F}$. Thus, we have

$$V_i(\mathcal{W}(\mathcal{M})) = V_i(\mathcal{W}(\mathcal{M} \setminus \mathcal{F})) + V_i(\mathcal{F}) \tag{2.23}$$

Now, we provide an upper bound on the value of $V_i(\mathcal{W}(\mathcal{M} \setminus \mathcal{F}))$. As a warm up, one can trivially prove an upper bound of $2\epsilon_i(2n - 1 - |B_1| - 2|B_2|)$ on $V_i(\mathcal{W}(\mathcal{M} \setminus \mathcal{F}))$. This follows from the fact that two sets of items are assigned to any agent and hence we have a total of $2n$ disjoint sets. Among these $2n$ sets, at least one of them is empty (since $g_i = \varnothing$) and at least $|B_1| + 2|B_2|$ of the sets contain a single blue item. On the other hand, by Lemmas 2.5.23, 2.5.29, 2.5.37 and 2.5.39 every set with white items is of worth at most $2\epsilon_i$ to a_i. Therefore, the total value of the white items in $\mathcal{M} \setminus \mathcal{F}$ to a_i is less than

$2\epsilon_i(2n-1-|B_1|-2|B_2|)$ and thus

$$V_i(\mathcal{W}(\mathcal{M}\setminus\mathcal{F})) \le 2\epsilon_i(2n-1-|B_1|-2|B_2|).$$

However, in order to complete the proof, we need a stronger upper bound on $V_i(\mathcal{W}(\mathcal{M}\setminus\mathcal{F}))$. To this end, we provide the following auxiliary lemma.

Lemma 2.5.46 *Let a_j be an agent such that $|f_j| = 1$ and $V_i(f_j) > V_i(f_i)$. Then, $V_i(g_j) < \epsilon_i$.*

Proof. First, note that if a_j is not satisfied yet, then $g_j = \varnothing$ and therefore $V_i(g_j) < \epsilon_i$. Otherwise, we argue that agent a_j is either satisfied in the second phase, or in the refinement phases of $\mathcal{C}_1$ or $\mathcal{C}_2$.

Consider the case that a_j is satisfied in the second phase. If $a_j \in \mathcal{S}_2^s \cup \mathcal{S}_3$, then by Lemma 2.5.39, $V_i(g_j) < \epsilon$ holds. Also, if $a_j \in \mathcal{S}_1^s$, considering the fact that a_i envies a_j, $a_i \prec_{pr} a_j$. Thus, by Lemma 2.5.39, we have $V_i(g_j) < \epsilon_i$.

Next, consider the case that a_j is in $\mathcal{S}_1^r \cup \mathcal{S}_2^r$. Note that the matching of the refinement phase of $\mathcal{C}_1$ preserves the property described in Lemma 2.5.17. Hence, if a_j belongs to $\mathcal{S}_1^r$, then either $a_j \prec_{pr} a_i$ or there is no edge between y_i and $M_1(y_j)$ in G_1, where $M_1(y_j)$ is the vertex matched with y_j in M_1. If $a_j \prec_{pr} a_i$, according to Observation 2.5.13, $V_i(f_j) \le 3/4 - \epsilon_i$ holds. On the other hand, by the definition, if no edge exists between y_i and $M_1(y_j)$ in G_1, $V_i(g_j) < \epsilon_i$. In addition to this, if a_j belongs to $\mathcal{S}_2^r$, according to Lemma 2.5.29, $V_i(g_j) < \epsilon_i$ holds. Therefore, Lemma 2.5.46 holds for the agents in $\mathcal{S}_1^r \cup \mathcal{S}_2^r$.

□

Note that since matching M of $G_{1/2}$ for building Cluster $\mathcal{C}_1$ is MCMWM according to condition (iii) of Lemma 2.5.14, there exists no agent a_k, such that $|g_k| = 1$ and $V_i(g_k) > 3/4 - \epsilon_i$. Otherwise, by assigning the item in g_j to a_i instead of the item in f_i, we can increase the total weight of the matching, that contradicts the maximality of M.

According to Lemma 2.5.46, for all the agents a_j with the property that f_j is a blue item that belongs to a partition in $\hat{B}_1$, $V_i(g_j) < \epsilon_i$ holds. The number of such agents is at least $|\hat{B}_1|$. Therefore, the total value of $V_i(\mathcal{W}(\mathcal{M} \setminus \mathcal{F}))$ is less than $2\epsilon_i \cdot (2n - 1 - |B_1| - 2|B_2| - |\hat{B}_1|) + \epsilon_i \cdot |\hat{B}_1|$. Combining the bounds obtained in Observation 2.22 and Lemma 2.5.46 by Inequality (2.23), we have:

$$V_i(\mathcal{F}) \geq 2\epsilon_i \cdot (2n - 2|B_2| - |B_1| - |\hat{B}_1|) + \epsilon_i \cdot (|\hat{B}_1|) - 2\epsilon_i \cdot (2n - 1 - |B_1| - 2|B_2| - |\hat{B}_1|) - \epsilon_i \cdot |\hat{B}_1|$$

That is:

$$V_i(\mathcal{F}) \geq 2\epsilon_i$$

This contradicts the fact that the set $\mathcal{F}$ is not feasible for a_i.

□

Finally, we show that all the agents in Cluster $\mathcal{C}_2$ are satisfied by the algorithm.

Lemma 2.5.47 *At the end of the algorithm we have $\mathcal{C}_2 = \varnothing$.*

The proof of Lemma 2.5.47 is a similar to both proofs of Lemmas 2.5.41 and 2.5.45. Let a_i be winner of Cluster $\mathcal{C}_2$. We consider two cases separately. (i) $\epsilon_i \geq 1/8$ and (ii) $\epsilon_i < 1/8$. In case $\epsilon_i \geq 1/8$, we use a similar argument to the proof of Lemma 2.5.41 and show $\mathcal{F}$ is feasible for a_i. If $\epsilon_i < 1/8$ we again use a coloring argument, but this time we color the items with 4 different colors. Again, via a double counting argument we show $\mathcal{F}$ is feasible for a_i and hence every agent of $\mathcal{C}_2$ is satisfied when the algorithm terminates.

Proof of Lemma 2.5.47: Lemmas 2.5.41 and 2.5.45 state that at the end of the algorithm, $\mathcal{C}_1 = \mathcal{C}_3 = \varnothing$. Now, let a_i be a winner of $\mathcal{C}_2$. We consider two cases separately: $\epsilon_i \geq 1/8$ and $\epsilon_i < 1/8$.

If $\epsilon_i \geq 1/8$, the proof follows from a similar argument we used to prove Lemma 2.5.45.

Lemma 2.5.48 *If $\epsilon_i \geq 1/8$, then the following inequality holds:*

$$\sum_{a_j \in \mathcal{S}} V_i(f_j \cup g_j) \leq |\mathcal{S}| + 1/8.$$

Proof. We know $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3$. For every agent a_j in $\mathcal{S}_3$, by Lemmas 2.5.33 and 2.5.37, we know $V_i(f_j) < 3/4$. Also, according to Lemma 2.5.39, $V_i(g_j) < \epsilon_i \leq 1/4$. Therefore,

$$\sum_{a_j \in \mathcal{S}_3} V_i(f_j \cup g_j) \leq \sum_{a_j \in \mathcal{S}_3} (3/4 + 1/4) = |\mathcal{S}_3|. \tag{2.24}$$

Now, consider an agent $a_j \in \mathcal{S}_1$. Note that by Lemma 2.5.15, $V_i(f_j) < 1/2$. Also, remark that either $a_j \in \mathcal{S}_1^r$ or $a_j \in \mathcal{S}_1^s$. If $a_j \in \mathcal{S}_1^r$ then according to Lemma 2.5.20, $V_i(g_j) < 1/2$ holds and hence $V_i(f_j \cup g_j) < 1$. Also, If $a_j \in \mathcal{S}_1^s$, then according to Lemma 2.5.39, $V_i(g_j) < 2\epsilon_i < 1/2$. Thus, in both cases, $V_i(f_j \cup g_j) < 1$ and hence:

$$\sum_{a_j \in \mathcal{S}_1} V_i(f_j \cup g_j) \leq \sum_{a_j \in \mathcal{S}_1} 1 = |\mathcal{S}_1|. \tag{2.25}$$

Finally consider a satisfied agent $a_j \in \mathcal{S}_2$. Again, remark that either $a_j \in \mathcal{S}_2^r$ or

$a_j \in \mathcal{S}_2^s$ holds.

Consider the case that $a_j \in \mathcal{S}_2^s$. If $a_j \prec_{pr} a_i$, then by Observation 2.5.13, $V_i(f_j) \leq 3/4 - \epsilon_i$ and by Lemma 2.5.39, $V_i(g_j) < 2\epsilon_i \leq 1/4 + \epsilon_i$ which means $V_i(f_j \cup g_j) < 1$. Moreover, if $a_i \prec_{pr} a_j$, according to Lemmas 2.5.33 and 2.5.39, $V_i(f_j \cup g_j) < 3/4 + \epsilon_i \leq 1$. Thus, we have:

$$\sum_{a_j \in \mathcal{S}_2^s} V_i(f_j \cup g_j) \leq \sum_{a_j \in \mathcal{S}_2^s} 1 = |\mathcal{S}_1| \tag{2.26}$$

It only remains to investigate the case where $a_j \in \mathcal{S}_2^r$. Note that since a_i is not satisfied in the refinement phase of $\mathcal{C}_2$, if $a_i \prec_{pr} a_j$, then $V_i(g_j) < \epsilon_i \leq 1/4$. Otherwise, we could assign the item in g_j to a_i in the refinement phase of $\mathcal{C}_2$. Also, by Lemma 2.5.33, $V_i(f_j) < 3/4$ holds which yields $V_i(f_j \cup g_j) < 1$.

Finally, if $a_j \prec_{pr} a_i$, by Observation 2.5.13 $V_i(f_j) \leq 3/4 - \epsilon_i$ holds. Corollary 2.5.25 states that there is at most one item b_k with $\mathcal{M}_k \in \mathcal{X}' \setminus \mathcal{X}'_{1/2}$ and $V_i(b_k) \geq 3/8$. Also, note that since b_k belongs to $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$, $V_i(\{b_k\}) < 1/2$ holds. For agent a_j, let b_l be the item that is assigned to a_j in the refinement of $\mathcal{C}_2$, i.e., $g_j = \{b_l\}$. We have

$$V_i(f_j \cup g_j) \leq 3/4 - \epsilon_i + V_i(\{b_l\}).$$

If $b_l \neq b_k$, $V_i(f_j \cup g_j) \leq 3/4 - \epsilon_i + 3/8$ holds which by the fact that $\epsilon_i \geq 1/8$, implies $V_i(f_j \cup g_j) \leq 3/4 - 1/8 + 3/8 \leq 1$. In addition to this, If $b_l = b_k$, $V_i(f_j \cup g_j) \leq 3/4 - 1/8 + 4/8 \leq 1 + 1/8$. But this can happen for at most one agent. Thus, for every agent a_j in $\mathcal{S}_2^r$, $V_i(f_j \cup g_j) \leq 1$ holds and for at most one agent $a_j \in \mathcal{S}_2^r$, $V_i(f_j \cup g_j) \leq 1 + 1/8$.

Thus, we have

$$\sum_{a_j \in \mathcal{S}_2^r} V_i(f_j \cup g_j) \leq |\mathcal{S}_2^r| + 1/8. \tag{2.27}$$

Inequality (2.27) together with Inequality (2.26) yields

$$\sum_{a_j \in \mathcal{S}_2} V_i(f_j \cup g_j) \leq |\mathcal{S}_2| + 1/8. \tag{2.28}$$

Furthermore, by Inequalities (2.24), (2.25) and (2.28) we have

$$\begin{aligned}\sum_{a_j \in \mathcal{S}} V_i(f_j \cup g_j) &= \sum_{a_j \in \mathcal{S}_1} V_i(f_j \cup g_j) + \sum_{a_j \in \mathcal{S}_2} V_i(f_j \cup g_j) + \sum_{a_j \in \mathcal{S}_3} V_i(f_j \cup g_j) \\ &\leq |\mathcal{S}_1| + |\mathcal{S}_2| + 1/8 + |\mathcal{S}_3| \\ &\leq |\mathcal{S}| + 1/8.\end{aligned} \tag{2.29}$$

□

By Lemma 2.5.48, value of agent a_i for the items assigned to the satisfied agents is less than $|\mathcal{S}| + 1/8$. Recall that $\mathcal{C}_2 = \mathcal{C}_3 = \varnothing$ and hence $|\mathcal{S}| = n - |\mathcal{C}_2|$. Therefore,

$$\sum_{a_j \in \mathcal{S}} V_i(f_j \cup g_j) \leq n - |\mathcal{C}_2| + 1/8. \tag{2.30}$$

Since a_i is a winner of $\mathcal{C}_2$, for all $a_j \in \mathcal{C}_2$, we have $V_i(f_j) \leq V_i(f_i)$. On the other hand, since the total value of all items for a_i is equal to n we have

$$
\begin{aligned}
V_i(\mathcal{F}) &= V_i(\mathcal{M}) - \sum_{a_j \in \mathcal{C}_2} V_i(f_j) - \sum_{a_j \in \mathcal{S}} V_i(f_j \cup g_j) \\
&= n - \sum_{a_j \in \mathcal{C}_2} V_i(f_j) - \sum_{a_j \in \mathcal{S}} V_i(f_j \cup g_j) \\
&\geq n - \sum_{a_j \in \mathcal{C}_2} V_i(f_j) - \left[n - |\mathcal{C}_2| + 1/8\right] \\
&= |\mathcal{C}_2| - 1/8 - \sum_{a_j \in \mathcal{C}_2} V_i(f_j).
\end{aligned}
\tag{2.31}
$$

Also, $V_i(f_i) = 3/4 - \epsilon_i$ holds and $V_i(f_j) \leq V_i(f_i)$ for any $a_j \in \mathcal{C}_2$ follows from the fact that a_i is a winner of $\mathcal{C}_2$. Therefore by Inequality (2.31) we have

$$
\begin{aligned}
V_i(\mathcal{F}) &\geq |\mathcal{C}_2| - 1/8 - \sum_{a_j \in \mathcal{C}_2} V_i(f_j) \\
&\geq |\mathcal{C}_2| - 1/8 - \sum_{a_j \in \mathcal{C}_2} V_i(f_i) \\
&= |\mathcal{C}_2| - 1/8 - |\mathcal{C}_2| V_i(f_i) \\
&= |\mathcal{C}_2| - 1/8 - |\mathcal{C}_2|(3/4 - \epsilon_i) \\
&= |\mathcal{C}_2|(1/4 + \epsilon_i) - 1/8.
\end{aligned}
$$

Recall that by the assumption $\epsilon_i \geq 1/8$ holds. Moreover, $\epsilon_i \leq 1/4$, and thus

$$
\begin{aligned}
V_i(\mathcal{F}) &\geq |\mathcal{C}_2|(1/4 + \epsilon_i) - 1/8 \\
&\geq |\mathcal{C}_2| 2\epsilon_i - 1/8 \\
&\geq |\mathcal{C}_2| 2\epsilon_i - \epsilon_i
\end{aligned}
$$

and since $|\mathcal{C}_2| \geq 1$,

$$\begin{aligned} V_i(\mathcal{F}) &\geq |\mathcal{C}_2| 2\epsilon_i - \epsilon_i \\ &\geq 2\epsilon_i - \epsilon_i \\ &\geq \epsilon_i \end{aligned}$$

and thus $\mathcal{F}$ is feasible for a_i. This contradicts the termination of the algorithm.

Next, we investigate the case where $\epsilon < 1/8$. Our proof for this case is similar to the one for $\mathcal{C}_1$. Let $\mathcal{S}_1^r$ be the agents in $\mathcal{S}_1$ that are satisfied in the refinement phase and let

$$\mathcal{M}_1^r = \bigcup_{a_j \in \mathcal{S}_1^r} f_j \cup g_j.$$

Lemma 2.5.21 states that the maxmin value of the agents in $\mathcal{C}_2 \cup \mathcal{C}_3$ for the items in $\mathcal{M}' = \mathcal{M} \setminus \mathcal{M}_1^r$ is at least 1. More precisely for every $a_j \in \mathcal{C}_2$:

$$\mathsf{MMS}_j^{n-|\mathcal{S}_1^r|}(\mathcal{M} \setminus \mathcal{M}_1^r) \geq 1 \tag{2.32}$$

We color the items of $\mathcal{M}'$ in one of four colors blue, red, green, or white. Initially, all the items are colored in white. For each agent $a_j \in \mathcal{N} \setminus \mathcal{S}_1^r$, if $|f_j| = 1$, then we color the item in f_j in blue. Also, if $|f_j| = 2$ (which means f_j is corresponding to a merged vertex), color both the elements of f_j in red. In addition to this, if $|g_j| = 1$ then color the item in g_j in green. For any set $S \subseteq \mathcal{M}$, we denote the subset of blue, red, green, and white items in S by $\mathcal{B}(S)$,$\mathcal{R}(S)$, $\mathcal{G}(S)$, and $\mathcal{W}(S)$, respectively. Recall that by Lemma 2.5.24, every pair of items in red or green are worth less that $3/4$ in total to a_i. In other words,

$$V_i(\{b_j, b_k\}) \leq 3/4.$$

for any two different items $b_j, b_k \in \mathcal{B}(\mathcal{M}) \cup \mathcal{G}(\mathcal{M})$. Also, according to Lemmas 2.5.39 and 2.5.37, every set including white items is worth less than $2\epsilon_i < 1/4$ to a_i.

Now, let $n' = n - |\mathcal{S}_1^r|$. Let $\mathcal{P} = \langle P_1, P_2, \dots, P_{n'} \rangle$ be the optimal n'−partitioning of $\mathcal{M}'$ for a_i. Recall that by Inequality (2.32) the value of every partition in $\mathcal{P}$ is at least 1 for a_i. Based on the number of blue and red items in every partition, we define three sets of partitions:

- B_{00} : Partitions with no red or blue items.
- B_{10} : Partitions with blue items, but without any red items.
- B_{01} : Partitions that contain at least one red item.

Next we prove Lemmas 2.5.49 and 2.5.50 to be used later in the proof.

Lemma 2.5.49 *Let $|\mathcal{G}(B_{00})|$ be the number of green items in the partitions of B_{00}. Then,*

$$V_i(\mathcal{W}(B_{00})) \geq (3|B_{00}| - |\mathcal{G}(B_{00})|) \cdot 1/4.$$

Proof. Let B_{00}^j be the set of partitions in B_{00} that contain exactly j green items. We have:

$$|\mathcal{G}(B_{00})| = \sum_{1 \leq j < \infty} j|B_{00}^j| \geq |B_{00}^1| + 2|B_{00}^2| + \sum_{3 \leq j < \infty} 3|B_{00}^j| \tag{2.33}$$

Also, we have:

$$3|B_{00}| = \sum_{0 \le j < \infty} 3|B_{00}^j| = 3|B_{00}^0| + 3|B_{00}^1| + 3|B_{00}^2| + \sum_{3 \le j < \infty} 3|B_{00}^j| \tag{2.34}$$

Finally, we argue that the value of white items in B_{00} is at least $|B_{00}^0| + |B_{00}^1| \cdot 1/2 + |B_{00}^3| \cdot 1/4$. This follows from the fact that every green item in $P_k \in B_{00}^1$ has a value less than $1/2$ and by Lemma 2.5.24, every pair of green items in $P_k \in B_{00}^2$ are worth less than $3/4$ to a_j. According to the fact that the value of every partition P_k is at least 1, we have:

$$V_i(\mathcal{W}(B_{00})) \ge |B_{00}^0| + |B_{00}^1| \cdot 1/2 + |B_{00}^3| \cdot 1/4 = \big(4|B_{00}^0| + 2|B_{00}^1| + |B_{00}^2|\big) \cdot 1/4 \tag{2.35}$$

According to Equations (2.33) and (2.34), we have:

$$3|B_{00}| - |\mathcal{G}(B_{00})| \le 3|B_{00}^0| + 2|B_{00}^1| + |B_{00}^2| \le 4|B_{00}^0| + 2|B_{00}^1| + |B_{00}^2| \tag{2.36}$$

Next we combine Equations (2.35) and (2.36) to obtain:

$$V_i(\mathcal{W}(B_{00})) \ge \big(3|B_{00}| - |\mathcal{G}(B_{00})|\big) \cdot 1/4 \tag{2.37}$$

□

Lemma 2.5.50 $V_i(\mathcal{W}(B_{10})) \ge (2|B_{10}| - |\mathcal{B}(B_{10})| - |\mathcal{G}(B_{10})|) \cdot 1/4$

Proof. First, note that every partition in B_{10} contains at least one blue item. Let B_{10}^w be the partitions in B_{10} that contains exactly one blue item and no green item. The other items in each partition of B_{10}^{w}, are white. Since the problem is $3/4$-irreducible, the value of every blue item to a_i is less than $3/4$ and therefore we have:

$$V_i(\mathcal{W}(B_{10})) \geq |B_{10}^{w}| \cdot 1/4$$

or

$$4V_i(\mathcal{W}(B_{10})) \geq |B_{10}^{w}|. \tag{2.38}$$

Moreover, let $B_{10}^{\bar{w}} = B_{10} \setminus B_{10}^{w}$. Since every partition in B_{10} contains at least one blue item, every partition in $B_{10}^{\bar{w}}$ contains at least two items with colors blue or green. Thus, we have:

$$|\mathcal{G}(B_{10}^{\bar{w}})| + |\mathcal{B}(B_{10}^{\bar{w}})| \geq 2|B_{10}^{\bar{w}}| \tag{2.39}$$

Summing up Equations (2.38) and (2.39) results in

$$4V_i(\mathcal{W}(B_{10})) + |\mathcal{G}(B_{10}^{\bar{w}})| + |\mathcal{B}(B_{10}^{\bar{w}})| \geq 2|B_{10}^{\bar{w}}| + |B_{10}^{w}|$$

which means:

$$4V_i(\mathcal{W}(B_{10})) \geq 2|B_{10}^{\bar{w}}| - |\mathcal{G}(B_{10}^{\bar{w}})| - |\mathcal{B}(B_{10}^{\bar{w}})| + |B_{10}^{w}|. \tag{2.40}$$

Morover, we have $|\mathcal{B}(B_{10})| = |\mathcal{B}(B_{10}^{w})| + |\mathcal{B}(B_{10}^{\bar{w}})|$. According to the fact that every partition in B_{10}^{w} contains exactly one blue item, $|\mathcal{B}(B_{10}^{w})| = |B_{10}^{w}|$ and hence, $|\mathcal{B}(B_{10})| =$

$|B_{10}^{w}| + |\mathcal{B}(B_{10}^{\bar{w}})|$. By Equation (2.40), we have:

$$4V_i(\mathcal{W}(B_{10})) \geq 2|B_{10}^{\bar{w}}| - |\mathcal{G}(B_{10}^{\bar{w}})| - |\mathcal{B}(B_{10})| + |B_{10}^{w}| + |B_{10}^{w}|.$$

Finally by the fact that $2|B_{10}^{w}| + 2|B_{10}^{\bar{w}}| = 2|B_{10}|$, we have:

$$4V_i(\mathcal{W}(B_{10})) \geq 2|B_{10}| - |\mathcal{G}(B_{10}^{\bar{w}})| - |\mathcal{B}(B_{10})|$$

which is:

$$V_i(\mathcal{W}(B_{10})) \geq \big(2|B_{10}| - |\mathcal{B}(B_{10})| - |\mathcal{G}(B_{10}^{\bar{w}})|\big) \cdot 1/4$$

□

For the partitions in B_{01}, we construct a graph $G_{01}\langle V_{01}, E_{01}\rangle$, where every vertex $v_j \in V_{01}$ corresponds to a partition $P_j \in B_{01}$. Consider an agent a_j such that f_j consists of a pair of red items $b_k, b_{k'}$ and let $b_k \in P_l$ and $b_{k'} \in P_{l'}$. We add an edge $(v_l, v_{l'})$ to E_{01}. By the definition of B_{01}, $P_l, P_{l'} \in B_{01}$ holds. Note that b_k and $b_{k'}$ might belong to the same partition, i.e., $P_l = P_{l'}$. In this case, we add a loop to G_{01}. Furthermore, for every item $b_k \in \mathcal{B}(B_{01})$, we add a loop to the vertex v_l, where $b_k \in P_l$.

Next, define R_j as the set of partitions in B_{01}, such that the degree of their corresponding vertices in V_{01} are equal to j. In other words:

$$P_k \in R_j \iff d(v_k) = j$$

Next we prove Lemma 2.5.51.

Lemma 2.5.51 *For R_1, we have:*

$$V_i(\mathcal{W}(R_1)) \geq (2|R_1| - |\mathcal{G}(R_1)|) \cdot 1/4$$

Proof. Consider a partition $P_j \in R_1$. Since $d(v_j) = 1$, P_j contains exactly one red item and no blue item. Thus, other items in P_j are either green or white. We show that

$$|\mathcal{G}(P_j)| + 4.V_i(\mathcal{W}(P_j)) \geq 2. \tag{2.41}$$

First, argue that if $|\mathcal{G}(P_j)| \geq 2$, then Inequality (2.41) holds. Also, if $|\mathcal{G}(P_j)| = 0$, then $V_i(\mathcal{W}(P_j)) \geq 1/2$, because the value of the red item in P_j is less than $1/2$ (recall that all the red items correspond to the vertices in $\mathcal{X}' \setminus \mathcal{X}'_{1/2}$). This immediately implies the fact that $4.V_i(\mathcal{W}(P_j)) \geq 2$. Finally, if $|\mathcal{G}(P_j)| = 1$, then by Lemma 2.5.24, the total value of the green and red items in P_j is less than $3/4$ and hence, $V_i(\mathcal{W}(P_j)) \geq 1/4$ which means $|\mathcal{G}(P_j)| + 4.V_i(\mathcal{W}(P_j)) \geq 2$.

Since Inequality (2.41) holds for every partition $P_j \in R_1$, we have:

$$\sum_{P_j \in R_1} \big(|\mathcal{G}(P_j)| + 4.V_i(\mathcal{W}(P_j))\big) \geq 2|R_1|$$

Therefore,

$$|\mathcal{G}(R_1)| + 4.V_i(\mathcal{W}(R_1)) \geq 2|R_1|$$

and hence,

$$V_i(\mathcal{W}(R_1)) \geq (2|R_1| - |\mathcal{G}(R_1)|) \cdot 1/4$$

$\square$

Lemma 2.5.52 *For R_2, we have:*

$$V_i(\mathcal{W}(R_2)) \geq (|R_2| - |\mathcal{G}(R_2)|) \cdot 1/4$$

Proof. Let P_j be a partition in R_2. First, we show the following inequality holds:

$$4V_i(\mathcal{W}(P_j)) + |\mathcal{G}(P_j)| \geq 1 \tag{2.42}$$

By the definition of R_2, degree of v_j is 2. Therefore, P_j contains two red items. Note that the degree of the partitions in B_{01} that contain blue items is at least 3. Thus, P_j contains no blue items. By Lemma 2.5.24, the total value of the red items in P_j is less than $3/4$. The rest of the items in P_j are either green or white. If P_j contains a green item, then Inequality (2.42) holds. On the other hand, if P_j contains no green items, then $V_i(\mathcal{W}(P_j)) \geq 1/4$ and hence, $4V_i(\mathcal{W}(P_j)) \geq 1$. Therefore, Inequality (2.42) holds in both cases.

Summing up Inequality (2.42) for all the partitions in R_2, we have:

$$\sum_{P_j \in R_2} 4V_i(\mathcal{W}(P_j)) + |\mathcal{G}(P_j)| \geq |R_2|$$

which means:

$$4V_i(\mathcal{W}(R_2)) + |\mathcal{G}(R_2)| \geq |R_2|$$

That is:

$$V_i(\mathcal{W}(R_2)) \geq \big(|R_2| - |\mathcal{G}(R_2)|\big) \cdot 1/4$$

□

Putting together Lemmas 2.5.49,2.5.50,2.5.51, and 2.5.52 we obtain the following lower bound on the valuation of a_i for all white items:

$$\begin{aligned}
V_i(\mathcal{W}(\mathcal{M}')) &= V_i(\mathcal{W}(B_{00})) + V_i(\mathcal{W}(B_{01})) + V_i(\mathcal{W}(B_{10})) \\
&\geq \Big(3|B_{00}| - |\mathcal{G}(B_{00})|\Big) \cdot 1/4 + \Big(2|B_{10}| - |\mathcal{B}(B_{10})| - |\mathcal{G}(B_{10})|\Big) \cdot 1/4 \\
&\quad + \Big(2|R_1| - |\mathcal{G}(R_1)|\Big) \cdot 1/4 + \Big(|R_2| - |\mathcal{G}(R_2)|\Big) \cdot 1/4 \\
&= \Big((3|B_{00}| + 2|B_{10}| + 2|R_1| + |R_2| - |\mathcal{B}(B_{10})|) - \big(|\mathcal{G}(B_{00})| \\
&+ |\mathcal{G}(B_{10})| + |\mathcal{G}(R_1)| + |\mathcal{G}(R_2)|\big)\Big) \cdot 1/4 \\
&\geq \Big((3|B_{00}| + 2|B_{10}| + 2|R_1| + |R_2|) - |\mathcal{B}(B_{10})| - |\mathcal{G}(\mathcal{M}')|\Big) \cdot 1/4
\end{aligned} \tag{2.43}$$

where $|\mathcal{G}(\mathcal{M}')|$ is the total number of green items.

The items in $\mathcal{W}(\mathcal{M}')$ are either allocated to an agent during the second phase, or are still in $\mathcal{F}$. Let $\mathcal{W}_2$ be the white items that are allocated to an agent during the second phase. We have:

$$V_i(\mathcal{W}(\mathcal{M}')) = V_i(\mathcal{W}_2) + V_i(\mathcal{F}) \tag{2.44}$$

Now, we present an upper bound on the value of $V_i(\mathcal{W}_2)$. First, note that the number of agents in $\mathcal{S} \setminus \mathcal{S}_1^r$ is n'. Each of these n' agents has two sets f_j and g_j, that leaves us $2n'$

sets. Since $g_i = \varnothing$ we know that at least one of these sets is empty. Moreover, of all these $|\mathcal{G}(\mathcal{M}')|$ sets contain a single green item and $|\mathcal{B}(B_{10})| + |E_{01}|$ of the sets contain either a single blue item, or a pair of red items (recall that each edge of G_{01} refers to a blue item or a pair of red items). Therefore, the number of the sets that contain only white items is at most:

$$2n' - 1 - |\mathcal{G}(\mathcal{M}')| - |\mathcal{B}(B_{10})| - |E_{01}|$$

By Lemmas 2.5.39 and 2.5.37, the value of every set with white items to a_i is less than $2\epsilon_i < 1/4$ and hence:

$$V_i(\mathcal{W}_2) \leq (2n' - 1 - |\mathcal{G}(\mathcal{M}')| - |\mathcal{B}(B_{10})| - |E_{01}|) \cdot 1/4 \tag{2.45}$$

Subtracting the lower bound obtained for $V_i(\mathcal{W}(\mathcal{M}'))$ in (2.43) from the upper bound for $V_i(\mathcal{W}_2)$ in (2.45) gives us a lower bound on the value of $\mathcal{F}$:

$$\begin{aligned}
V_i(\mathcal{F}) &= V_i(\mathcal{W}(\mathcal{M}')) - V_i(\mathcal{W}_2) \\
&\geq \Big((3|B_{00}| + 2|B_{10}| - |\mathcal{B}(B_{10})| + 2|R_1| + |R_2|) - |\mathcal{G}(\mathcal{M}')| \Big) \cdot 1/4 - V_i(\mathcal{W}_2) \\
&\geq \Big((3|B_{00}| + 2|B_{10}| - |\mathcal{B}(B_{10})| + 2|R_1| + |R_2|) - |\mathcal{G}(\mathcal{M}')| \Big) \cdot 1/4 \\
&\quad - \Big(2n' - 1 - |\mathcal{G}(\mathcal{M}')| - |\mathcal{B}(B_{10})| - |E_{01}| \Big) \cdot 1/4 \\
&= \Big(3|B_{00}| + 2|B_{10}| + 2|R_1| + |R_2| - 2n' + 1 + |E_{01}| \Big) \cdot 1/4 \\
&= \Big(2|B_{00}| + 2|B_{10}| + |B_{00}| + |E_{01}| + 2|R_1| + |R_2| - 2n' + 1 \Big) \cdot 1/4
\end{aligned} \tag{2.46}$$

Next we provide Lemmas 2.5.53, 2.5.55, and 2.5.54 to complete the proof.

Lemma 2.5.53 $|B_{00}| \geq |E_{01}| - |B_{01}|$

Proof. First, note that $|B_{00}|+|B_{10}|+|B_{01}| = n'$. Moreover we have $|\mathcal{B}(B_{10})|+|E_{01}| \leq n'$. To show this Lemma, note that each edge in G_{01} corresponds to the first set of an agent in $\mathcal{S} \setminus \mathcal{S}_1^r$. Also, every blue item in B_{10} corresponds to the first set of an agent in $\mathcal{S} \setminus \mathcal{S}_1^r$. Therefore, the total number of the agents must be more than this number. By the definition of B_{10}, we know that $|\mathcal{B}(B_{10})| \geq |B_{10}|$. Therefore, we have:

$$\begin{aligned} |B_{00}| + |B_{10}| + |B_{01}| &\geq |\mathcal{B}(B_{10})| + |E_{01}| \\ &\geq |(B_{10})| + |E_{01}| \end{aligned} \tag{2.47}$$

This means:

$$|B_{00}| \geq |E_{01}| - |B_{01}|$$

$\square$

Lemma 2.5.54 $|E_{01}| \geq 3/2\sum_{j\geq 3}|R_j| + |R_2| + |R_1|/2$

Proof. $|E_{01}| = \frac{\sum_{v_j \in V_{01}} d(v_j)}{2} = \frac{\sum_j j|R_j|}{2} \geq 3/2\sum_{j\geq 3}|R_j| + |R_2| + |R_1|/2.$ $\square$

Lemma 2.5.55 $|B_{00}| \geq \frac{\sum_{j\geq 3}|R_j| - |R_1|}{2}$

Proof. By Lemma 2.5.53, $|B_{00}| \geq |E_{01}| - |B_{01}|$. Furthermore, by Lemma 2.5.54,

$$|E_{01}| \geq 3/2\sum_{j\geq 3}|R_j| + |R_2| + |R_1|/2.$$

By these two inequalities, we have:

$$|B_{00}| \geq 3/2 \sum_{j\geq 3} |R_j| + |R_2| + |R_1|/2 - |B_{01}| \tag{2.48}$$

Also, since there is a one-to-one correspondence between B_{01} and V_{01}, $|B_{01}| = |V_{01}|$ holds. By the definition of R_j, we have:

$$|V_{01}| = \sum_j |R_j| \tag{2.49}$$

By replacing the value obtained for B_{01} from (2.49) into Inequality (2.48), we have:

$$\begin{aligned} |B_{00}| &\geq 1/2 \sum_{j\geq 3} |R_j| - |R_1|/2 \\ &= \frac{\sum_{j\geq 3} |R_j| - |R_1|}{2}. \end{aligned} \tag{2.50}$$

□

By applying Lemmas 2.5.55 and 2.5.54 to Inequality (2.46) we have:

$$\begin{aligned} V_i(\mathcal{F}) &= \Big(2|B_{00}| + 2|B_{10}| + |B_{00}| + |E_{01}| + 2|R_1| + |R_2| - 2n' + 1\Big) \cdot 1/4 \\ &\geq \Big(2|B_{00}| + 2|B_{10}| + \frac{\sum_{j\geq 3} |R_j| - |R_1|}{2} \\ &+ 3/2 \sum_{j\geq 3} |R_j| + |R_2| + |R_1|/2 + 2|R_1| + |R_2| - 2n' + 1\Big) \cdot 1/4 \\ &= \Big(2|B_{00}| + 2|B_{10}| + \sum_{j\geq 3} 2|R_j| + 2|R_2| + 2|R_1| - 2n' + 1\Big) \cdot 1/4 \end{aligned}$$

Finally, note that $\sum_{j\geq 3} 2|R_j| + 2|R_2| + 2|R_1| = 2|V_{01}| = 2|B_{01}|$. This, together with the fact that $|B_{00}| + |B_{01}| + |B_{10}| = n'$, yields $V_i(\mathcal{F}) \geq (2n' - 2n' + 1) \cdot 1/4$. This means $V_i(\mathcal{F}) \geq 1/4$ which is a contradiction since $\mathcal{F}$ is feasible for a_i. □

Theorem 2.5.56 *All the agents are satisfied before the termination of the algorithm.*

Proof. By Lemmas 2.5.41, 2.5.45, and 2.5.47, at the end of the algorithm all agents are satisfied which means each has received a subset of items which is worth at least $3/4$ to him. □

2.5.5 Algorithm

In this section, we present a polynomial time algorithm to find a $(3/4 - \epsilon)$-MMS allocation in the additive setting. More precisely, we show that our method for proving the existence of a $3/4$-MMS allocation can be used to find such an allocation in polynomial time. Recall that our algorithm consists of two main phases: The clustering phase and the bag filling phase. In Sections 2.5.5.1 and 2.5.5.2 we separately explain how to implement each phase of the algorithm in polynomial time. Given this, there are still a few computational issues that need to be resolved. First, in the existential proof, we assume $\mathsf{MMS}_i = 1$ for every agent $a_i \in \mathcal{N}$. Second, we assume that the problem is $3/4$-irreducible. Both of these assumptions are without loss of generality for the existential proof due to Observation 2.2.2 and the fact that one can scale the valuation functions to ensure $\mathsf{MMS}_i = 1$ for every agent a_i. However, the computational aspect of the problem will be affected by these assumptions. The first issue can be alleviated by incurring an additional $1 + \epsilon$ factor to the approximation guarantee. Epstein and Levin [34] show that for a given additive function f, MMS^n_f can be approximated within a factor $1 + \epsilon$ for constant

ϵ in time $\mathsf{poly}(n)$. Thus, we can scale the valuation functions to ensure $\mathsf{MMS}_i = 1$ while losing a factor of at most $1+\epsilon$. Therefore, finding a $(3/4-\epsilon)$-MMS allocation can be done in polynomial time if the problem is $3/4$-irreducible. Finally, in Section 2.5.5.3 we show how to reduce the $3/4$-reducible instances and extend the algorithm to all instances of the problem. The algorithm along with the reduction yields Theorem 2.5.57

Theorem 2.5.57 *For any $\epsilon > 0$, there exists an algorithm that finds a $(3/4-\epsilon)$-MMS allocation in polynomial time.*

2.5.5.1 The Clustering Phase

Recall that in the clustering phase we cluster the agents into three sets $\mathcal{C}_1$,$\mathcal{C}_2$, and $\mathcal{C}_3$. In order to build Cluster $\mathcal{C}_1$, we find an MCMWM of the $1/2$-filtering of the value graph. This can be trivially done in polynomial time since finding an MCMWM is polynomially tractable [35]. However, the refinement phase of Cluster $\mathcal{C}_1$ requires finding $F_G(\mathcal{X}, M)$ for a giving graph G and a matching M. In what follows, we show this problem can also be solved in polynomial time.

Notice that finding an MCMWM of G can be done in polynomial time [35]. Therefore, in order to determine $F_H(M, \hat{\mathcal{X}})$, it only suffices to find the vertices of $\hat{\mathcal{X}}$ that are reachable from the unmatched vertices of $\hat{\mathcal{Y}}$ by an alternating path. Let $\hat{X}$ be the set of these vertices. We can find $\hat{X}$ using a depth-first-search from the unmatched vertices of $\hat{\mathcal{Y}}$. By definition, $F_H(M, \hat{\mathcal{X}}) = \hat{\mathcal{Y}} \setminus \hat{X}$. Therefore, $F_H(M, \hat{\mathcal{X}})$ can be found in polynomial time.

In addition to $F_G(\mathcal{X}, M)$, we also need to find a matching of the graph which satisfies the conditions of Lemma 2.5.17. We show in the following that this problem also can be solved in polynomial time. First, note that in Lemma 2.5.18 we prove that G_1 has a matching that saturates all the vertices of W_1. Now, let p_{a_k} be the position of a_k in the topological ordering of $\mathcal{C}_1$, as described in the proof of Lemma 2.5.17. Furthermore, Let M_1 be a matching that minimizes the following expression.

$$\sum_{(x_j, y_i) \in M_1} p_i.$$

Recall that in the proof Lemma 2.5.17, we show that M_1 satisfies the condition described in Lemma 2.5.17. Here, we show that M_1 can be found in polynomial time. To this end, we model this with a network design problem.

Orient every edge $(x_j, y_i) \in G_1$ from y_i to x_j and set the cost of this edge to p_{a_i}. Also, add a source node s and add a directed edge from s to every vertex of $V_{\mathcal{C}_1}$ with cost 0. Furthermore, add a sink node t and add directed edges from the vertices of W_1 to t with cost 0. Finally, set the capacity of all edges to 1.

One can observe that in a minimum cost maximum flow from s to t in this network, the edges with non-zero flow between $V_{\mathcal{C}_1}$ and W_1 form a maximum matching M_1. In addition to this, since the cost of the flow is minimal, $\sum_{(x_j, y_i) \in M_1} cost(x_j, y_i)$ is minimized. Therefore, in this matching, $\sum_{(x_j, y_i) \in M_1} p_i$ is minimized. Thus, the matching with desired properties of Lemma 2.5.17 can be found in polynomial time.

The same algorithms can be used to compute Cluster $\mathcal{C}_2$. Finally, we put the rest of the agents in Cluster $\mathcal{C}_3$.

2.5.5.2 The bag filling Phase

In each round of the second phase, we iteratively find a minimal feasible subset of $\mathcal{F}$ and allocate its items to the agent with the lowest priority in $\Phi(S)$. Note that for a feasible set S, one can trivially find the agent with lowest priority in $\Phi(S)$ in polynomial time. Thus, it only remains to show that we can find a minimal feasible subset of $\mathcal{F}$ in polynomial time.

Consider the following algorithm, namely *reverse bag filling algorithm*: Start with a bag containing all the items of $\mathcal{F}$ and so long as there exists an item b_j in the bag such that after removing b_j, the set of items in the bag is still feasible, remove b_j from the bag. After this process, the remaining items in the bag form a minimally feasible subset of $\mathcal{F}$. Therefore, this phase can be run in polynomial time.

2.5.5.3 Reducibility

The most challenging part of our algorithm is dealing with the $3/4$-irreducibility assumption. The catch is that, in order to run the algorithm, we don't necessarily need the $3/4$-irreducibility assumption. Recall that we leverage the following three consequences of irreducibility to prove the existential theorem.

- The value of every item in $\mathcal{M}$ is less that $3/4$ to every agent.
- Every pair of items in $\mathcal{X}'' \setminus \mathcal{X}''_{1/2}$ is in total worth less than $3/4$ to any agent.
- The condition of Lemma 2.5.18 holds.

Therefore, the algorithm works so long as the mentioned conditions hold. Note that, although it is not clear whether determining if an instance of the problem is $3/4$-reducible is polynomially tractable, all of the above conditions can be validated in polynomial time. This is trivial for the first two conditions; we iterate over all items or pairs of items and check if the condition holds for these items. The last condition, however, is harder to validate.

The condition of Lemma 2.5.18 holds if for all $S \subseteq W_1$, $|N(S)| > |S|$. Recall that in the proof of Lemma 2.5.18 we showed that if this condition does not hold, then $F_{G_1}(M, \mathcal{X})$ is non-empty. Next, we showed that if $F_{G_1}(M, \mathcal{X})$ is non-empty, then we can reduce the problem via satisfying every agents of $F_{G_1}(M, \mathcal{X})$ by his matched item in M. Therefore, on the computational side, we only need to find whether $F_{G_1}(M, \mathcal{X})$ is empty which indeed can be determined in polynomial time.

Note that every time we reduce the problem, $|\mathcal{N}|$ is decreased by at least 1, which implies the number of times we reduce the problem is no more than n. Moreover, our reduction takes a polynomial time. Thus, the running time of the algorithm is polynomial.

2.6 Submodular Agents

Previous work on the fair allocation problem was limited to the additive agents [12, 2]. In real-world, however, valuation functions are usually more complex than additive ones. As an example, imagine an agent is interested in at most k items. More precisely, he is indifferent between receiving k items or more than k items. Such a valuation function is called k-demand and cannot be modeled by additive functions. k-demand functions are a subclass of submodular set functions which have been extensively studied the literature of different contexts, e.g., optimization, mechanism design, and game theory [36, 37, 38, 39, 40, 41, 42, 43, 30].

In this section, we study the fair allocation problem where the valuations of agents are submodular. We begin by presenting an impossibility result; We show in Section 2.6.1 that the best guarantee that we can achieve for submodular agents is upper bounded by $3/4$. Next, we give a proof to the existence of a $1/3$-MMS allocation in this setting. This is followed by an algorithm that finds such an allocation in polynomial time. This is surprising since even finding the MMS of a submodular function is NP-hard and cannot be implemented in polynomial time unless P=NP [34]. In our algorithm, we assume we have access to query oracle for the valuation of agents; That is, for any set S and any agent a_i, $V_i(S)$ can be computed via a given query oracle in time $O(1)$.

2.6.1 Upper Bound

We begin by providing an upper bound. In this section, we show for some instances of the problem with submodular agents, no allocation can be better than $3/4$-MMS. Our counter-example is generic; We show this result for any number of agents.

Theorem 2.6.1 *For any $n \geq 2$, there exists an instance of the fair allocation problem with n submodular agents where no allocation is better than $3/4$-MMS.*

Proof. We construct an instance of the problem that does not admit any $3/4+\epsilon$-MMS allocation. To this end, let n be the number of agents and $\mathcal{M} = \{b_1, b_2, \ldots, b_m\}$ where $m = 2n$. Furthermore, let $f : 2^{\mathcal{M}} \rightarrow \mathbb{R}$ be as follows:

$$f(S) = \begin{cases} 0, & \text{if } |S| = \varnothing \\ 1, & \text{if } |S| = 1 \\ 2, & \text{if } |S| > 2 \\ 2, & \text{if } S = \{b_{2i}, b_{2i+1}\} \text{ for some } i \\ 3/2, & \text{if } |S| = 2 \text{ and } S \neq \{b_{2i}, b_{2i+1}\} \text{ for any } i. \end{cases}$$

Notice that $\mathsf{MMS}_f^n = 2$. Moreover, in what follows we show that f is submodular. To this end, suppose for the sake of contradiction that there exist sets S and S' such that $S \subseteq S'$ and for some element b_i we have:

$$f(S' \cup \{b_i\}) - f(S') > f(S \cup \{b_i\}) - f(S). \tag{2.51}$$

Since f is monotone and $S' \neq S$, $f(S' \cup \{b_i\}) - f(S') > 0$ holds and thus S' cannot have more than two items. Therefore, S' contains at most two items and thus S is either empty or contains a single element. If S is empty, then adding every element to S has the highest increase in the value of S and thus Inequality (2.51) doesn't hold. Therefore, S contains a single element and S' contains exactly two elements. Thus, $f(S) = 1$ and $f(S') \geq 3/2$. Therefore, f(S $\cup\{b_i\}) - f(S) \geq 1/2$ and $f(S' \cup \{b_i\}) - f(S') \leq 1/2$ which contradicts Inequality (2.51).

Now, for agents $a_1, a_2, \ldots, a_{n-1}$ we set $V_i = f$ and for agent a_n we set $V_n = f(\mathsf{inc}(S))$ where b_i is in $\mathsf{inc}(S)$ if and only if either $i > 1$ and $b_{i-1} \in S$ or $i = 1$ and $b_m \in S$.

The crux of the argument is that for any allocation of the items to the agents, someone receives a value of at most $3/2$. In case an agent receives fewer than two items, his valuation for his items would be at most 1. Similarly, if an agent receives more than two items, someone has to receive fewer than 2 items and the proof is complete. Therefore, the only case to investigate is where everybody receives exactly two items. We show in such cases, $\min V_i(A_i) = 3/2$ for all possible allocations. If all agents $a_1, a_2, \ldots, a_{n-1}$ receive two items whose value for them is exactly equal to 2, then by the construction of f, the value of the remaining items is also equal to 2 to them. Thus, a_n's valuation for the items he receives is equal to $3/2$. $\square$

Remark that one could replace function f with an XOS function

$$g(S) = \begin{cases} 0, & \text{if } |S| = \varnothing \\ 1, & \text{if } |S| = 1 \\ 2, & \text{if } |S| > 2 \\ 2, & \text{if } S = \{b_{2i}, b_{2i+1}\} \text{ for some } i \\ 1, & \text{if } |S| = 2 \text{ and } S \neq \{b_{2i}, b_{2i+1}\} \text{ for any } i. \end{cases}$$

and make the same argument to achieve a $1/2$-MMS upper bound for XOS and subadditive agents.

Theorem 2.6.2 *For any $n > 1$, there exists an instance of the fair allocation problem with n XOS agents where no allocation is better than $1/2$-MMS.*

2.6.2 Existential Proof

In this section we provide an existential proof to a $1/3$-MMS allocation. Due to the algorithmic nature of the proof, we show in Section 2.6.3 that such an allocation can be computed in time $\mathsf{poly}(n, m)$. For simplicity, we scale the valuation functions to ensure $\mathsf{MMS}_i = 1$ for every agent a_i.

We begin by introducing the ceiling functions.

Definition 2.6.3 *Given a set function $f(.)$, we define $f^x(.)$ as follows:*

$$f^x(S) = \begin{cases} f(S), & \text{if } f(S) \le x \\ x, & \text{if } f(S) > x. \end{cases}$$

A nice property of the ceiling functions is that they preserve submodularity, fractionally subadditivity, and sub-additivity.

Lemma 2.6.4 *For any real number $x \ge 0$, we have:*

1. *Given a submodular set function $f(.)$, $f^x(.)$ is submodular.*

2. *Given an XOS set function $f(.)$, $f^x(.)$ is XOS.*

3. *Given an subadditive set function $f(.)$, $f^x(.)$ is also subadditive.*

Observation 2.6.5 *$f^x(S) \le x$ for every given S.*

Observation 2.6.6 *$f^x(S) \le f(S)$ for every given S.*

Proof Of Lemma 2.6.4:

First Claim: By definition of submodular functions, for given sets A and B we have:

$$f(A \cup B) \leq f(A) + f(B) - f(A \cap B)$$

We prove that $f^x(.)$ is a submodular function in three different cases:

First Case: Let both $f(A)$ and $f(B)$ be at least x. According to Observation 2.6.5, $f^x(A \cup B)$ and $f^x(A \cap B)$ are bounded by x. Therefore, $f^x(A \cup B) + f^x(A \cap B) \leq 2x$, which yields:

$$f^x(A \cup B) + f^x(A \cap B) \leq f^x(A) + f^x(B)$$

Second Case: In this case one of $f(A)$ and $f(B)$ is at least x. We have $f(A \cup B) \geq x$ and $f(A \cap B)$ is no more than max $\{f(A), f(B)\}$. As a result $f^x(A \cup B)$ and one of $f^x(A)$ or $f^x(B)$ are equal to x which yields:

$$f^x(A \cup B) + f^x(A \cap B) \leq f^x(A) + f^x(B)$$

Third Case: In this case both $f(A)$ and $f(B)$ are less than x, and $f(A \cap B)$ is less than x too. Since $f^x(A) = f(A)$, $f^x(B) = f(B)$, $f^x(A \cap B) = f(A \cap B)$, according to Observation 2.6.6, $f^x(A \cup B) \leq f(A \cup B)$ holds. Since $f(.)$ is a submodular function, we conclude that:

$$f^x(A \cup B) \leq f^x(A) + f^x(B) - f^x(A \cap B).$$

Second Claim: Since $f(.)$ is an XOS set function, by definition, there exists a finite set of additive functions $\{f_1, f_2, \dots, f_\alpha\}$ such that

$$f(S) = \max_{i=1}^{\alpha} f_i(S)$$

for any set $S \subseteq \mathsf{ground}(f)$. With that in hand, for a given real number x, we define an XOS set function $g(.)$, and show $g(.)$ is equal to $f^x(.)$.

We define $g(.)$ on the same domain as $f(.)$. Moreover, based on $\{f_1, f_2, \dots, f_\alpha\}$, we define a finite set of additive functions $\{g_1, g_2, \dots, g_\beta\}$ that describe g. More precisely, for each set S in domain of $f(.)$ we define a new additive function like g_γ in $g(.)$ as follows: Without loss of generality let f_δ be the function which maximizes $f(S)$. For each $b_i \notin S$ let $g_\gamma(b_i) = 0$. Furthermore, for each $b_i \in S$ if $f(S) \leq x$ let $g_\gamma(b_i) = f_\delta(b_i)$, and otherwise let $g_\gamma(b_i) = \frac{x}{f(S)} f_\delta(b_i)$.

We claim that $g(.)$ is equivalent to $f^x(.)$, which implies $f^x(.)$ is an XOS function. $g(.)$ and $f^x(.)$ are two functions which have equal domains. First, we prove that $g(S) \leq f(S)$ for any given set S. According to construction of $g(.)$, for each additive function in $g(.)$ such g_γ, there is at least one additive function in $f(.)$ such f_δ where $g_\gamma(b_i) \leq f_\delta(b_i)$ for each $b_i \in \mathcal{M}$. Therefore, for any given set S we have:

$$g(S) \leq f(S) \tag{2.52}$$

Now, according to the construction of $g(.)$, for any given set S where $f(S) \leq x$, we have a function $g_\gamma(S) = f(S)$, and where $f(S) > x$, we have a function $g_\gamma(S) = x$. Therefore, we can conclude that:

$$g(S) \geq f^x(S) \tag{2.53}$$

For any given set S where $f(S) \leq x$, according to the definition of $f^x(.)$, $f(S) = f^x(S)$, and using Inequalities (2.52) and (2.53) we argue that $f^x(S) = g(S)$. Moreover, according to the construction of $g(.)$, $g(S) \leq x$ for any given set S. Therefore, for any given set S where $f(S) > x$, according to the definition of $f^x(.)$ and Inequality (2.53), $f^x(S) = g(S) = x$. As a result, by considering these two cases we argue that $f^x(.)$ and $g(.)$ are equivalent, which shows $f^x(.)$ is an XOS function.

Third Claim: In this claim, we use a similar argument to the first claim. By definition of subadditive functions for any given sets A and B, we have:

$$f(A \cup B) \leq f(A) + f(B)$$

We prove that $f^x(.)$ meets the definition of subadditive functions by considering two different cases. In the first case at least one of $f(A)$ and $f(B)$ is at least x, and in the second case both $f(A)$ and $f(B)$ is less than x.

First Case: In this case $f^x(A) + f^x(B)$ is at least x, and since $f^x(S) \leq x$ for any given set S, $f^x(A \cup B) \leq x$. Therefore,

$$f^x(A \cup B) \leq f^x(A) + f^x(B)$$

Second Case: Since $f^x(A \cup B) \leq f(A \cup B)$, $f(A \cup B) \leq f(A) + f(B)$, $f(A) = f^x(A)$, and $f(B) = f^x(B)$, we have:

$$f^x(A \cup B) \leq f^x(A) + f^x(B)$$

□

The idea behind the existence of a $1/3$-MMS allocation is simple: Suppose the problem is $1/3$-irreducible and let $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$ be an allocation of items to the agents that maximizes the following expression:

$$\sum_{a_i \in \mathcal{N}} V_i^{2/3}(A_i) \tag{2.54}$$

We refer to Expression (2.54) by $\mathsf{ex}^{(2/3)}(\mathcal{A})$. We prove $V_i(A_i) \geq 1/3$ for every agent $a_i \in \mathcal{N}$. By the reducibility principal, it only suffices to show every $1/3$-irreducible instance of the problem admits a $1/3$-MMS allocation. The main ingredients of the proof are Lemmas 2.5.1, 2.6.7 and 2.6.8.

Lemma 2.6.7 *Let $S_1, S_2, \ldots, S_k$ be k disjoint sets and $f_1, f_2, \ldots, f_k$ be k submodular functions. We remove an element e from $\bigcup S_i$ uniformly at random to obtain sets $S_1^* = S_1 \setminus \{e\}, S_2^* = S_2 \setminus \{e\}, \ldots, S_k^* = S_k \setminus \{e\}$. In this case we have*

$$\mathbb{E}[\sum f_i(S_i^*)] \geq \sum f_i(S_i) \frac{|\bigcup S_i| - 1}{|\bigcup S_i|}.$$

The high-level intuition behind the proof of Lemma 2.6.7 is as follows: For submodular functions, the smaller the size of a set is, the higher the marginal values for adding items to that set will be. Based on that, we show the summation of marginal decreases for removing each element is bounded by the total value of the set and that completes the proof.

Proof Of Lemma 2.6.7: Since $f(.)$ is submodular, according to the definition of submodular functions, for every given sets X and Y in domain of $f(.)$ with $X \subseteq Y$ and every $x \in \mathcal{M} \setminus Y$ we have:

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y) \tag{2.55}$$

Let $S_i = \{e_1, e_2, \ldots, e_\alpha\}$, $T_0 = \varnothing$, and $T_j = \{e_1, e_2, \ldots, e_j\}$, for every $1 \leq j \leq \alpha$. Since $T_j \subseteq S_i$ for each $0 \leq j \leq \alpha$ and f_i is a submodular function, according to Inequality (2.55) we have:

$$\sum_{1 \leq j \leq \alpha} f_i(S_i \setminus T_{j-1}) - f_i(S_i \setminus T_j) \geq \sum_{1 \leq j \leq \alpha} f_i(S_i) - f_i(S_i - e_j) \tag{2.56}$$

Since $f_i(S_i) = \sum_{1 \leq j \leq \alpha} f_i(S_i \setminus T_{j-1}) - f_i(S_i \setminus T_j)$, we can rewrite Inequality (2.56) for every $1 \leq i \leq k$ as follows:

$$f_i(S_i) \geq \sum_{e \in S_i} f_i(S_i) - f_i(S_i - e) \tag{2.57}$$

For every $1 \le i \le k$ we can rewrite Inequality (2.57) as follows:

$$\sum_{e \in s_i} f_i(S_i - e) \ge (|S_i| - 1) f_i(S_i) \tag{2.58}$$

By adding $(|\bigcup S_i| - |S_i|) f_i(S_i)$ to the both sides of Inequality (2.58), we have:

$$\begin{aligned} (|\bigcup S_i| - |S_i|) f_i(S_i) + \sum_{e \in S_i} f_i(S_i - e) &= \sum_{e \in \bigcup S_i} f_i(S_i \setminus \{e\}) \\ &\ge (|\bigcup S_i| - 1) f_i(S_i) \end{aligned} \tag{2.59}$$

Since Inequality (2.59) holds for every $1 \le i \le k$, we can sum up both sides of Inequality (2.59) as follows:

$$\sum_{1 \le i \le k} \sum_{e \in \bigcup S_i} f_i(S_i - e) \ge \sum_{1 \le i \le k} (|\bigcup S_i| - 1) f_i(S_i) \tag{2.60}$$

By dividing both sides of Inequality (2.60) over $1/|\bigcup S_i|$ we obtain:

$$\begin{aligned} \frac{1}{|\bigcup S_i|} \Big(\sum_{e \in \bigcup S_i} \sum_{1 \le i \le k} f_i(S_i - e) \Big) &= \mathbb{E}\Big[\sum_{1 \le i \le k} f_i(S_i^*) \Big] \\ &\ge \sum_{1 \le i \le k} f_i(S_i) \frac{|\bigcup S_i| - 1}{|\bigcup S_i|}. \end{aligned} \tag{2.61}$$

□

Lemma 2.6.8 *Let f be a submodular function and $S_1, S_2, \ldots, S_k$ be k disjoint sets such that $f(S_i) \ge 1$ for every set S_i. Moreover, let $S \subseteq \bigcup S_i$ be a set such that $f(S) < 1/3$. If* *we pick an element $\{e\}$ of $\bigcup S_i \setminus S$ uniformly at random, we have:*

$$\mathbb{E}[f(S \cup \{e\}) - f(S)] \geq \frac{2k/3}{|\bigcup S_i \setminus S|}.$$

The proof of Lemma 2.6.8 is very similar to that of Lemma 2.6.7. The main point is that in submodular functions, the marginal increase decreases as the sizes of sets grow.

Proof Of Lemma 2.6.8: Similar to the proof of Lemma 2.6.7, we use Inequality (2.55) as a definition of submodular functions. Let $S'_i = S_i \setminus S = \{e_1, e_2, \ldots, e_\alpha\}$, $T_0 = S$, and $T_j = S \cup \{e_1, e_2, \ldots, e_j\}$ for $1 \leq j \leq \alpha$. According to $f(S) < 1/3$, $f(S \cup S'_i) \geq 1$, and Inequality (2.55) as a definition of sub-modular functions, we have:

$$\begin{aligned} 2/3 &< f(S \cup S') - f(S) \\ &= \sum_{1 \leq j \leq \alpha} f(T_{j-1} \cup \{e_j\}) - f(T_{j-1}) \\ &\leq \sum_{e \in S'_i} f(S \cup \{e\}) - f(S) \end{aligned} \tag{2.62}$$

Similar to Inequality (2.60), we can rewrite Inequality (2.62) with a summation, since Inequality (2.62) holds for any $1 \leq i \leq k$.

$$2k/3 < \sum_{1 \leq i \leq k} \sum_{e \in S'_i} f(S \cup \{e\}) - f(S) \tag{2.63}$$

By dividing both sides of Inequality (2.63) over $1/|\bigcup S_i \setminus S|$ we have:

$$\frac{2k/3}{|\bigcup S_i \setminus S|} < \frac{1}{|\bigcup S_i \setminus S|}(\sum_{1\le i\le k}\sum_{e\in S'_i} f(S\cup\{e\}) - f(S)) \tag{2.64}$$
$$= \mathbb{E}[f(S\cup\{e\}) - f(S)]$$

□

Next, we show the fair allocation problem with submodular agents admits a $1/3$-MMS allocation[3].

Theorem 2.6.9 *The fair allocation problem with submodular agents admits a* $1/3$-MMS *allocation.*

Proof. By Lemma 2.2.2, the problem boils down to the case of $1/3$-irreducible instances. Let the problem be $1/3$-irreducible and $\mathcal{A}$ be an allocation that maximizes $\mathsf{ex}^{(2/3)}$. Suppose for the sake of contradiction that $V_i(A_i) < 1/3$ for some agent a_i. In this case we select an item b_r from $\mathcal{M} \setminus A_i$ uniformly at random to create a new allocation $\mathcal{A}^r$ as follows:

$$A^r_j = \begin{cases} A_j \setminus \{b_r\}, & \text{if } i \neq j \\ A_j \cup \{b_r\} & \text{if } i = j. \end{cases}$$

In the rest we show $\mathbb{E}[\mathsf{ex}^{(2/3)}(\mathcal{A}^r)] > \mathsf{ex}^{(2/3)}(\mathcal{A})$ which contradicts the maximality

[3]Almost one year after the first draft of our work, the existense of a $1/10$-MMS allocation in the submodular case along with an algorithm to find a $1/31$ approximation algorithm for the submodular case is also proved in [19]. They also study the problem in the additive setting and present another $2/3$-MMS algorithm. This work is completely parallel to and independent of our paper. Moreover, their analysis is fundamentally different from our analysis and also their bounds are looser.

of $\mathcal{A}$. Note that by Lemma 2.6.7 the following inequality holds:

$$\mathbb{E}[\sum_{j\neq i} V_j^{2/3}(A_j^r)] \geq \sum_{j\neq i} V_j^{2/3}(A_j)\frac{|\mathcal{M}\setminus A_i|-1}{|\mathcal{M}\setminus A_i|}. \tag{2.65}$$

Moreover, by Lemma 2.6.8 we have

$$\mathbb{E}[V_i(A_i^r) - V_i(A_i)] \geq \frac{2n/3}{|\mathcal{M}\setminus A_i|}. \tag{2.66}$$

Inequality (2.65) along with Inequality (2.66) shows

$$\begin{aligned}
\mathbb{E}[\mathsf{ex}^{(2/3)}(\mathcal{A}^r)] &= \mathbb{E}[\sum_{j\neq i} V_j^{2/3}(A_j^r)] + \mathbb{E}[V_i(A_i^r)] \\
&\geq \sum_{j\neq i} V_j^{2/3}(A_j)\frac{|\mathcal{M}\setminus A_i|-1}{|\mathcal{M}\setminus A_i|} + \mathbb{E}[V_i(A_i^r)] \\
&\geq \sum_{j\neq i} V_j^{2/3}(A_j)\frac{|\mathcal{M}\setminus A_i|-1}{|\mathcal{M}\setminus A_i|} + \frac{2n/3}{|\mathcal{M}\setminus A_i|} + V_i(A_i) \\
&\geq \sum_{j\neq i} V_j^{2/3}(A_j)\frac{|\mathcal{M}\setminus A_i|-1}{|\mathcal{M}\setminus A_i|} + \frac{2n/3}{|\mathcal{M}\setminus A_i|} + V_i^{(2/3)}(A_i) \\
&\geq \sum_{j\neq i} V_j^{2/3}(A_j)\frac{|\mathcal{M}\setminus A_i|-1}{|\mathcal{M}\setminus A_i|} + \frac{2n/3}{|\mathcal{M}\setminus A_i|} + V_i^{(2/3)}(A_i)\frac{|\mathcal{M}\setminus A_i|-1}{|\mathcal{M}\setminus A_i|} \\
&= \mathsf{ex}^{(2/3)}(\mathcal{A})\frac{|\mathcal{M}\setminus A_i|-1}{|\mathcal{M}\setminus A_i|} + \frac{2n/3}{|\mathcal{M}\setminus A_i|}.
\end{aligned} \tag{2.67}$$

Recall that by Lemma 2.5.1, the value of agent a_i for any item alone is bounded by $1/3$ and thus $\mathbb{E}[V_i(A_i^r) - V_i(A_i)] = \mathbb{E}[V_i^{2/3}(A_i^r) - V_i^{2/3}(A_i)]$. Notice that by the definition, $V_j^{(2/3)}$ is always bounded by $2/3$ and also $V_i(A_i) < 1/3$, therefore, $\mathsf{ex}^{(2/3)}(\mathcal{A}) \leq 2n/3 - 1/3$

and thus

$$\begin{aligned}\mathbb{E}[\mathsf{ex}^{(2/3)}(\mathcal{A}^r)] &\geq \mathsf{ex}^{(2/3)}(\mathcal{A})\frac{|\mathcal{M}\setminus A_i|-1}{|\mathcal{M}\setminus A_i|} + \frac{2n/3}{|\mathcal{M}\setminus A_i|} \\ &\geq \mathsf{ex}^{(2/3)}(\mathcal{A}) + \frac{1/3}{|\mathcal{M}\setminus A_i|} \\ &\geq \mathsf{ex}^{(2/3)}(\mathcal{A}) + 1/3m.\end{aligned} \tag{2.68}$$

□

2.6.3 Algorithm

In this section we give an algorithm to find a $1/3$-MMS allocation for submodular agents. We show our algorithm runs in time $\mathsf{poly}(n,m)$.

For simplicity, we assume for every agent a_i, MMS_i is given as input to the algorithm. However, computing MMS_i alone is an NP-hard problem. That said, we show in Section 2.7.2.2 that such a computational barrier can be lifted by a combinatorial trick. We refer the reader to Section 2.7.2.2 for a more detailed discussion. The procedure is illustrated in Algorithm 4: Based on Theorem 2.6.9, one can show that in every iteration of the algorithm value of $\mathsf{ex}^{2/3}(\mathcal{A})$ is increased by at least $1/3m$. Moreover, such an element b_e can be easily found by iterating over all items in time $O(m)$. Furthermore, the number of iterations of the algorithm is bounded by $2nm$, since $\mathsf{ex}^{2/3}(\mathcal{A})$ is bounded by $2n/3$. Therefore, Algorithm 4 finds a $1/3$-MMS allocation in time $\mathsf{poly}(n,m)$.

Theorem 2.6.10 *Given access to query oracles, one can find a* $1/3$-MMS *allocation for submodular agents in polynomial time.*

As a corollary of Theorem 2.6.10, one can show that the problem of finding the

Algorithm 4: Finding a $1/3$-MMS allocation for submodular agents

```
Data: $\mathcal{N}, \mathcal{M}, \langle V_1, V_2, \ldots, V_n\rangle, \langle \mathsf{MMS}_1, \mathsf{MMS}_2, \ldots, \mathsf{MMS}_n\rangle$
1  For every $a_j$, scale $V_j$ to ensure $\mathsf{MMS}_j = 1$;
2  while there exist an agent $a_i$ and an item $b_j$ such that $V_i(\{b_j\}) \geq 1/3$ do
3      Allocate $\{b_j\}$ to $a_i$;
4      $\mathcal{M} = \mathcal{M} \setminus b_j$;
5      $\mathcal{N} = \mathcal{N} \setminus a_i$;
6  end
7  $\mathcal{A}$ = an arbitrary allocation of the items to the agents;
8  while $\min V_j^{2/3}(A_j) < 1/3$ do
9      $i$ = the agent who receives the lowest value in allocation $\mathcal{A}$;
10     Find an item $b_e$ such that:
       $\mathsf{ex}(\langle A_1 \setminus \{b_e\}, A_2 \setminus \{b_e\}, \ldots, A_{i-1} \setminus \{b_e\}, A_i \cup \{b_e\}, A_{i+1} \setminus \{b_e\}, \ldots, A_n \setminus \{b_e\}\rangle) \geq$
       $\mathsf{ex}(\mathcal{A}) + 1/3m$;
11     $\mathcal{A} = \langle A_1 \setminus \{b_e\}, A_2 \setminus \{b_e\}, \ldots, A_{i-1} \setminus \{b_e\}, A_i \cup \{b_e\}, A_{i+1} \setminus \{b_e\}, \ldots, A_n \setminus \{b_e\}\rangle$;
12 end
13 For every $a_i \in \mathcal{N}$ allocate $A_i$ to $a_i$;
```

maxmin value of a submodular function admits a 3 approximation algorithm.

Corollary 2.6.11 *For a given submodular function f, we can in polynomial time split the elements of ground set into n dijsoint sets $S_1, S_2, \ldots, S_n$ such that*

$$f(S_i) \geq \mathsf{MMS}_f^n/3$$

for every $1 \leq i \leq n$.

2.7 XOS Agents

Class of fractionally subadditive (XOS) set functions is a super class of submodular functions. These functions too, have been subject of many studies in recent years [44, 45, 26, 46, 47, 48, 49, 29, 50]. Similar to sub-modular functions, in this section we show a $1/5$-MMS allocation is possible when all agents have XOS valuations. Furthermore, we complement our proof by providing a polynomial algorithm to find a $1/8$-MMS allocation in Section 2.7.2.

2.7.1 Existential Proof

In this section we show every instance of the fair allocation problem with XOS agents admits a $1/5$-MMS allocation. Without loss of generality, we assume $\mathsf{MMS}_i = 1$ for every agent a_i. Recall the definition of ceiling functions.

Definition 2.7.1 *Given a set function $f(.)$, we define $f^x(.)$ as follows:*

$$f^x(S) = \begin{cases} f(S), & \text{if } f(S) \leq x \\ x, & \text{if } f(S) > x. \end{cases}$$

As stated in Lemma 2.6.4, for every XOS function and every real number $x \geq 0$, f^x is also XOS. The proof of this section is similar to the result of Section 4. However, the details are different since XOS functions do not adhere to the nice structure of submodular functions. For every allocation $\mathcal{B}$, we define $\mathsf{ex}^{2/5}(\mathcal{B})$ as follows:

$$\mathsf{ex}^{2/5}(\mathcal{B}) = \sum_{a_i \in \mathcal{N}} V_i^{2/5}(B_i).$$

Now Let $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$ be an allocation of items to the agents that maximizes $\mathsf{ex}^{2/5}$. Provided that the problem is $1/5$-irreducible, we show $\mathcal{A}$ is a $1/5$-MMS allocation. Before we proceed to the main proof, we state Lemmas 2.7.2, and 2.7.3 as auxiliary observations.

Lemma 2.7.2 *Let $f(.)$ be an XOS set function and $f(S) = \beta$ for a set $S \subseteq \mathsf{ground}(f)$. If we divide S into k (possibly empty) sets $S_1, S_2, \ldots, S_k$ then*

$$\sum_{i=1}^{k} \Big(f(S) - f(S \setminus S_i) \Big) \leq f(S).$$

Roughly speaking, the proof follows from the fact that for at least one of the additive set functions in the representation of f, we have $g_j(S) = \beta$. The rest of the proof is trivial by the additive properties of g_j.

Proof of Lemma 2.7.2: According to the definition of XOS function, $f(.)$ is an XOS function with a finite set of additive functions $\{g_1, g_2, \ldots, g_\alpha\}$ where $f(S) = \max_{i=1}^{\alpha} g_i(S)$ for any set $S \in \mathsf{ground}(f)$. Let $g_j(.)$ be the additive function which maximizes S. Let $g_j(S_1) = \alpha_1, g_j(S_2) = \alpha_2, \ldots, g_j(S_k) = \alpha_k$, which yields $\beta = \sum \alpha_i$. Since $g_j(S_i) = \alpha_i$, $f(S \setminus S_i) \geq \beta - \alpha_i$. Therefore, we have:

$$\begin{aligned} \sum f(S) - f(S \setminus S_i) &\leq \sum \beta - (\beta - \alpha_i) \\ &= \beta \\ &= f(S) \end{aligned} \tag{2.69}$$

□

By Lemma 2.5.1, we know that in every $1/5$-irreducible instance of the problem, the value of every item for a person is bounded by $1/5$.

For XOS functions, we again, leverage the reducibility principal to show another important property of the $1/5$-irreducible instances of the problem.

Lemma 2.7.3 *In a $1/5$-irreducible instance of the problem, for a given agent a_i we can divide the items into $2n$ sets $S_1, S_2, \ldots, S_{2n}$ such that*

$$V_i(S_i) \geq 2/5$$

for every $1 \leq i \leq 2n$.

Proof. According to the definition of MMS, we know that a_i can divide items to n sets $\mathcal{P} = \langle P_1, P_2, \ldots, P_n \rangle$ such that $V_i(P_j) \geq 1$ for any P_j. The catch is that a_i can divide each of these n sets to two disjoint sets such that the value of each of these new sets be at least $2/5$ to him. Let $T = \{b_1, b_2, \ldots, b_\gamma\}$ be one of these n sets, and $g_j(.)$ be an additive function which maximizes $V_i(T)$. Let $T_k = \{b_1, b_2, \ldots, b_k\}$ for any $1 \leq k \leq \gamma$. According to Lemma 2.5.1, since the problem is $1/5$-irreducible, the value of any item is less than $1/5$ to a_i. Therefore, there is a set T_k among T_1 to T_γ where $2/5 \leq g_j(T_k) < 3/5$. Since $g_j(.)$ is one of additive functions of XOS function V_i, we have $V_i(T_k) \geq 2/5$. Moreover, since $g_j(T_k) < 3/5$, $g_j(T \setminus T_k) \geq 2/5$, which yields $V_i(T \setminus T_k) \geq 2/5$. As a conclusion, we can divide each of n sets to two disjoint sets with at least $2/5$ value to a_i.

□

We first apply Lemma 2.5.1 and show in such instances of the problem the valuation of every agent for every item is bounded by $1/5$. We remark that for every agent a_i, one can split the items into n partitions such that each partition is worth at least 1 to a_i. Combining the two observations, we conclude that such a decomposition is possible for every agent a_i. Next we prove the main theorem of this section.

Theorem 2.7.4 *The fair allocation problem with XOS agents admits a* $1/5$-MMS *allocation.*

Proof. Similar to what we did in Section 2.5, we only prove this for $1/5$-irreducible instances of the problem. By Observation 2.2.2, we can extend this result to all instances of the problem.

Consider an allocation $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$ of items to the agents that maximizes $\mathsf{ex}^{2/5}$. We show that such an allocation is $1/5$-MMS. Suppose for the sake of contradiction that there exists an agent a_i who receives a set of items which are together of worth less than $1/5$ to him. More precisely,

$$V_i^{2/5}(A_i) = V_i(A_i) < 1/5.$$

Since the problem is $1/5$-irreducible, by Lemma 2.7.3, we can divide the items into $2n$ sets $S_1, S_2, \ldots, S_{2n}$ such that $V_i(S_j) \geq 2/5$ for every $1 \leq j \leq 2n$. Note that in this case, $V_i^{2/5}(S_j) = 2/5$ follows from the definition. Moreover by monotonicity, $V_i^{2/5}(S_j \cup A_i) = 2/5$ holds for every j.

Now consider $2n$ allocations $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^{2n}$ such that

$$\mathcal{A}^j = \langle A_1^j, A_2^j \ldots, A_n^j \rangle$$

for every $1 \leq j \leq 2n$ where

$$A_k^j = \begin{cases} A_k \cup S_j, & \text{if } k = i \\ \\ A_k \setminus S_j, & \text{if } k \neq i. \end{cases}$$

We show at least one of these allocations has a higher for $\mathsf{ex}^{2/5}$ than $\mathcal{A}$. Since $V_i^{2/5}$ is XOS, by Lemma 2.7.2 we have

$$\sum_{j=1}^{2n} \Big(V_k^{2/5}(A_k) - V_k^{2/5}(A_k \setminus S_j) \Big) \leq V_k^{2/5}(A_j)$$

for every $a_k \neq a_i$ and thus

$$\begin{aligned} \sum_{j=1}^{2n} V_k^{2/5}(A_k^j) &= \sum_{j=1}^{2n} V_k^{2/5}(A_j \setminus S_j) \\ &\geq 2n V_k^{2/5}(A_k) - V_k^{2/5}(A_k) \\ &= (2n-1) V_k^{2/5}(A_k) \end{aligned} \tag{2.70}$$

Moreover, since $V_i^{2/5}(A_i) < 1/5$, we have

$$\begin{aligned} \sum_{a_j \neq a_i} V_j^{2/5}(A_j) &> \sum_{a_j \in \mathcal{N}} V_j^{2/5}(A_j) - 1/5 \\ &= \mathsf{ex}^{2/5}(\mathcal{A}) - 1/5. \end{aligned} \tag{2.71}$$

Furthermore, since $V_i^{2/5}(S_j \cup A_i) = 2/5$ for every $1 \leq j \leq 2n$, we have

$$\begin{aligned} \sum_{a_k \neq a_i} V_k^{2/5}(A_k^j) &= \sum_{a_k \in \mathcal{N}} V_k^{2/5}(A_k^j) - 2/5 \\ &= \mathsf{ex}^{2/5}(\mathcal{A}^j) - 2/5 \end{aligned} \tag{2.72}$$

Finally, by combining Inequalities (2.70), (2.71), and (2.72) we have

$$
\begin{aligned}
\sum_{j=1}^{2n} \mathsf{ex}^{2/5}(\mathcal{A}^j) &= \sum_{j=1}^{2n}(2/5 + \sum_{a_k \neq a_i} V_k^{2/5}(A_k^j)) \\
&= 4n/5 + \sum_{j=1}^{2n} \sum_{a_k \neq a_i} V_k^{2/5}(A_k^j) \\
&\geq 4n/5 + \sum_{a_k \neq a_i} (2n-1) V_k^{2/5}(A_k) \\
&\geq 4n/5 + (2n-1)(\mathsf{ex}^{2/5}(\mathcal{A}) - 1/5) \\
&\geq 2n \cdot \mathsf{ex}^{2/5}(\mathcal{A}) + (4n - 2n + 1)/5 - \mathsf{ex}^{2/5}(\mathcal{A}) \\
&\geq 2n \cdot \mathsf{ex}^{2/5}(\mathcal{A}) + (2n+1)/5 - \mathsf{ex}^{2/5}(\mathcal{A})
\end{aligned}
$$

Now notice that since $V_k^{2/5}(A_k) \leq 2/5$, we have

$$
\begin{aligned}
\mathsf{ex}^{2/5}(\mathcal{A}) &= \sum_{k=1}^{n} V_k^{2/5}(A_k) \\
&\leq \sum_{k=1}^{n} 2/5 \\
&\leq 2n/5.
\end{aligned}
$$

and thus

$$
\begin{aligned}
\sum_{j=1}^{2n} \mathsf{ex}^{2/5}(\mathcal{A}^j) &\geq 2n \cdot \mathsf{ex}^{2/5}(\mathcal{A}) + (2n+1)/5 - \mathsf{ex}^{2/5}(\mathcal{A}) \\
&\geq 2n \cdot \mathsf{ex}^{2/5}(\mathcal{A}) + (2n+1)/5 - 2n/5 \\
&\geq 2n \cdot \mathsf{ex}^{2/5}(\mathcal{A}) + 1/5.
\end{aligned}
$$

Therefore, $\mathsf{ex}^{2/5}(\mathcal{A}^j) > \mathsf{ex}^{2/5}(\mathcal{A}) + 1/10n$ holds for at least one $\mathcal{A}^j$ which contradicts the maximality of $\mathcal{A}$. □

2.7.2 Algorithm

In this section we provide a polynomial time algorithm for finding a $1/8$-MMS allocation for the fair allocation problem with XOS agents. The algorithm is based on a similar idea that we argued for the proof of Theorem 2.7.4. Remark that our algorithm only requires access to demand and XOS oracles. It does *not* have any additional information about the maxmin values. This makes the problem computationally harder since computing the maxmin values is NP-hard [34]. We begin by giving a high-level intuition of the algorithm and show the computational obstacles can be overcome by combinatorial tricks. Consider the pseudo-code described in Algorithm 5.

Algorithm 5: Algorithm for finding a $1/8$-MMS allocation

```
   Data: $\mathcal{N}, \mathcal{M}, \langle V_1, V_2, \dots, V_n \rangle$
1  For every $a_j$, scale $V_j$ to ensure $\mathsf{MMS}_j = 1$;
2  while there exist an agent $a_i$ and an item $b_j$ such that $V_i(\{b_j\}) \geq 1/8$ do
3      Allocate $\{b_j\}$ to $a_i$;
4      $\mathcal{M} = \mathcal{M} \setminus b_j$;
5      $\mathcal{N} = \mathcal{N} \setminus a_i$;
6  end
7  $\mathcal{A}$ = an arbitrary allocation of the items to the agents;
8  while $\min V_j^{1/4}(A_j) < 1/8$ do
9      $i$ = the agent who receives the lowest value in allocation $\mathcal{A}$;
10     Find a set $S$ such that:
       $\mathsf{ex}^{1/4}(\langle A_1 \setminus S, A_2 \setminus S, \dots, A_{i-1} \setminus S, A_i \cup S, A_{i+1} \setminus S, \dots, A_n \setminus S \rangle) \geq$
       $\mathsf{ex}^{1/4}(\mathcal{A}) + 1/12n$;
11     $\mathcal{A} = \langle A_1 \setminus S, A_2 \setminus S, \dots, A_{i-1} \setminus S, A_i \cup S, A_{i+1} \setminus S, \dots, A_n \setminus S \rangle$;
12 end
13 For every $a_i \in \mathcal{N}$ allocate $A_i$ to $a_i$;
```

As we show in Section 2.7.2.1, Command 10 of the algorithm is always doable.

More precisely, there always exists a set S that holds in the condition of Command 10. Notice that in every step of the algorithm, $\mathsf{ex}^{1/4}(\mathcal{A})$ is increased by at least $1/12n$ and this value is bounded by $1/4 \cdot n = n/4$. Therefore the algorithm terminates after at most $3n^2$ steps and the allocation is guaranteed to be $1/8$-MMS.

That said, there are two major computational obstacles in the way of running Algorithm 5. Firstly, finding a set S that holds in the condition of Command 10 can not be trivially done in polynomial time. Second, scaling the valuation functions to ensure $\mathsf{MMS}_i = 1$ for all agents is NP-hard and cannot be done in polynomial time unless P=NP. To overcome the former, in Section 2.7.2.1 we provide an algorithm for finding such a set S in polynomial time. Next, in Section 2.7.2.2, we present a combinatorial trick to run the algorithm in polynomial time without having to deal with NP-hardness of scaling the valuation functions.

2.7.2.1 Executing Command 10 in Polynomial Time

In this section we present an algorithm to execute Command 10 of Algorithm 5. We show that such a procedure can be implemented via demand oracles.

Let for every $b_j \notin A_i$, c_j be the amount of contribution that b_j makes to $\mathsf{ex}^{1/4}(\mathcal{A})$. We set $p_e = 3(n/(n-1))c_e$ and ask the demand oracle of V_i to find a set S that maximizes $V_i(S) - \sum_{b_j \in S} p_j$. Via a trivial calculation, one can show that $V_i(S) - \sum_{b_j \in S} p_j \geq 1/4$ holds for at least one set of items. The reason this is correct is that one can divide the items into n partitions where each is worth at least 1 to a_i. Moreover, the summation of prices for the items is bounded by $3n/(n-1) \cdot (\sum_{j \neq i} V_j^{1/4}(A_j)) \leq 3n/4$. Therefore, for at least one of those partitions $V_i(S) - \sum_{b_j \in S} p_j$ is at least $1/4$. Thus, the set that the oracle reports is worth at least $1/4$ to a_i.

Now, let S^* be the set that the oracle reports and for every $b_j \in S^*$, c^*_j be the contribution of b_j to $V_i(S^*)$. We sort the items of S^* based on $c^*_j - p_j$ in non-increasing order. Next, we start with an empty bag and add the items in their order to the bag until the total value of the items in the bag to a_i reaches $1/4$. Since the value of every item alone is bounded by $1/8$, the total value of the items in the bag to a_i is bounded by $3/8$. Thus the contribution of those items to $\mathsf{ex}^{1/4}(\mathcal{A})$ is at most $(3/8)/(3n/(n-1)) \leq 1/8 - 1/(10n)$. Therefore, removing items of the bag from other allocations and adding them to A_i, increases $\mathsf{ex}^{1/4}(\mathcal{A})$ by at least $1/10n$.

Remark that one can use the same argument to prove this even if $\mathsf{MMS}_i \geq 1/(1 + 1/10n)$.

2.7.2.2 Running Algorithm 5 in Polynomial Time

As aforementioned, scaling valuation functions to ensure $\mathsf{MMS}_i = 1$ for every agent a_i is an NP-hard problem since determining the maxmin values is hard even for additive agents [2]. Therefore, unlike Section 2.7.2.1, in this section we massage the algorithm to make it executable in polynomial time.

Suppose an oracle gives us the maxnmin values of the agents. Provided that we can run Command 10 of Algorithm 5 in polynomial time, we can find a $1/8$-MMS allocation in polynomial time. Therefore, in case the oracle reports the actual maxmin values, the solution is trivial. However, what if the oracle has an error in its calculations? There are two possibilities: (i) Algorithm 5 terminates and finds an allocation which is $1/8$-MMS with respect to the reported maxmin values. (ii) The algorithm fails to execute Command 10, since no such set S holds in the condition of Command 10. The intellectual merit of this section boils down to investigation of the case when algorithm fails to execute Command 10. We show, this only happens due to an overly high misrepresentation of the maxmin value for agent a_i. Note that a_i is the agent who receives the lowest value in the last cycle of the execution.

Observation 2.7.5 *Given $\langle d_1, d_2, \dots, d_n\rangle$ as an estimate for the maxmin values, if Algorithm 5 fails to execute Command 10 for an agent a_i, then we have*

$$d_i \geq (1 + 1/10n)\mathsf{MMS}_i.$$

Proof of Observation 2.7.5 follows from the argument of Section 2.7.2.1. More precisely, as mentioned in Section 2.7.2.1, such a set S exists, if $\mathsf{MMS}_i \geq 1/(1 + 1/10n)$. Thus, given that the procedure explained in Section 2.7.2.1 fails to find such a set, one can conclude the the reported value for MMS_i is at least $(1/(1 + 1/10n))$ times its actual value. Based on Observation 2.7.5, we propose Algorithm 6 for implementing a maxmin oracle.

Note that in the beginning of the algorithm, we set $d_i = V_i(\mathcal{M})$ which is indeed greater than or equal to MMS_i. By Lemma 2.7.5, every time we decrease the value of d_i for an agent a_i, we preserve the condition $d_i \geq \mathsf{MMS}_i$ for that agent. Therefore, in every step of the algorithm, we have $d_i \geq \mathsf{MMS}_i$ and thus the reported allocation which is $1/8$-MMS with respect to d_i's is also $1/8$-MMS with respect to true maxmin values.

Algorithm 6: Implementing a maxmin oracle

```
Data: $\mathcal{N}, \mathcal{M}, \langle V_1, V_2, \ldots, V_n \rangle$
1  for every $a_i \in \mathcal{N}$ do
2      $d_i \leftarrow V_i(\mathcal{M})$;
3  end
4  while true do
5      Run Algorithm 5 assuming maxmin values are $d_1, d_2, \ldots, d_n$;
6      if the Algorithm fails to run Command 10 for an agent $a_i$ then
7          $d_i \leftarrow d_i/(1 + 1/10n)$;
8      end
9      else
10         Report the allocation and terminate the algorithm;
11     end
12 end
```

Thus, the algorithm provides a correct $1/8$-MMS allocation in the end. All that remains is to show the running time of the algorithm is polynomial.

Notice that every time we decrease d_i for an agent a_i, we multiply this value by $1/(1 + 1/10n)$, hence the number of such iterations is polynomial in n, unless the valuations are super-exponential in n. Since we always assume the input numbers are represented by $\mathsf{poly}(n)$ bits, the number of iterations is bounded by $\mathsf{poly}(n)$ and hence the algorithm terminates after a polynomial number of steps.

Theorem 2.7.6 *Given access to demand and XOS oracles, there exists a polynomial time algorithm that finds a $1/8$-MMS allocation for XOS agents.*

An elegant consequence of Theorem 2.7.6 is a 8-approximation algorithm for determining the maxmin value of an XOS function with r partitions.

Corollary 2.7.7 *Given an XOS function f, an integer number r, and access to demand and XOS oracles of f, there exists a 8-approximation polytime algorithm for determining MMS_f^r.*

Proof. We construct an instance of the fair allocation problem with r agents, all of whom have a valuation function equal to f. We find a $1/8$-MMS allocation of the items to the agents in polynomial time and report the minimum value that an agent receives as output.

The $1/8$ guarantee follows from the fact that every agent receives a subset of values that are worth $1/8$-MMS_i to him, and since MMS_i is exactly equal to MMS^r_f, every partition has a value of at least $\mathsf{MMS}^r_f/8$. $\square$

Remark 2.7.8 *A similar procedure can also be used to overcome the challenge of computing the maxmin values for the algorithm described in Section 2.6.3.*

2.8 Subadditive Agents

In this section we present a reduction from subadditive agents to XOS agents. More precisely, we show for every subadditive set function $f(.)$, there exists an XOS function $g(.)$, where g is dominated by f but the maxmin value of g is within a logarithmic factor of the maxmin value of f. We begin by an observation. Suppose we are given a subadditive function f on set $\mathsf{ground}(f)$, and we wish to approximate f with an additive function g which is dominated by f. In other words, we wish to find an additive function g such that

$$\forall S \subseteq \mathsf{ground}(f) \qquad g(S) \leq f(S)$$

and $g(\mathsf{ground}(f))$ is maximized. One way to formulate g is via a linear program. Suppose $\mathsf{ground}(f) = \{b_1, b_2, \ldots, b_m\}$ and let $g_1, g_2, \ldots, g_m$ be m variables that describe g in the following way:

$$\forall S \subseteq \mathsf{ground}(f) \qquad g(S) = \sum_{b_i \in S} g_i.$$

Based on this formulation, we can find the optimal additive function g by LP 2.73.

$$\begin{aligned} \text{maximize:} \quad & \sum_{b_i \in \mathsf{ground}(f)} g_i && && (2.73) \\ \text{subject to:} \quad & \sum_{b_i \in S} g_i \leq f(S) && \forall S \subseteq \mathsf{ground}(f) \\ & g_i \geq 0 && \forall b_i \in \mathsf{ground}(f) \end{aligned}$$

We show the objective function of LP 2.73 is lower bounded by $f(\mathsf{ground}(f))/\log m$. The basic idea is to first write the dual program and then based on a probabilistic method, lower bound the optimal value of the dual program by $f(\mathsf{ground}(f))/\log m$.

Lemma 2.8.1 *The optimal solution of LP 2.73 is at least $f(\mathsf{ground}(f))/\log m$.*

Proof. To prove the lemma, we write the dual of LP 2.73 as follows:

$$\begin{aligned} \text{minimize:} \quad & \sum_{S \subseteq \mathsf{ground}(f)} \alpha_S f(S) && && (2.74) \\ \text{subject to:} \quad & \sum_{S \ni b_i} \alpha_S \geq 1 && \forall b_i \in \mathsf{ground}(f) \\ & \alpha_S \geq 0 && \forall S \subseteq \mathsf{ground}(f) \end{aligned}$$

By the strong duality theorem, the optimal solutions of LP 2.73 and LP 2.74 are equal [51]. Next, based on the optimal solution of LP 2.74, we define a randomized procedure to draw a set of elements: We start with an empty set S^* and for every set $S \subseteq \mathsf{ground}(f)$ we add *all* elements of S to S^* with probability α_S. Since f is subadditive, the marginal increase of $f(S^*)$ by adding elements of a set S to S^* is bounded by $f(S)$ and thus the expected value of $f(S^*)$ is bounded by the objective of LP 2.74. In other words:

$$\mathbb{E}[f(S^*)] \leq \sum_{S \subseteq \mathsf{ground}(f)} \alpha_S f(S) \tag{2.75}$$

Remark that we repeat this procedure for all subsets of $\mathsf{ground}(S)$ independently and thus for every $b_i \in \mathsf{ground}(f)$, $\sum_{S \ni b_i} \alpha_S \geq 1$ holds we have

$$\mathsf{PR}[b_i \in S^*] \geq 1 - 1/e \simeq 0.632121 > 1/2 \tag{2.76}$$

for every element $b_i \in \mathsf{ground}(s)$. Now, with the same procedure, we draw $\lceil \log m \rceil + 2$ sets $S^*_1, S^*_2, \ldots, S^*_{\lceil \log m \rceil + 2}$ *independently*. We define $\hat{S} = \bigcup S^*_i$. By Inequality (2.76) and the union bound we show

$$\begin{aligned}
\mathsf{PR}[\hat{S} = \mathsf{ground}(f)] &\geq 1 - \sum_{b_i \in \mathsf{ground}(i)} \mathsf{PR}[b_i \notin \hat{S}] \\
&= 1 - \sum_{b_i \in \mathsf{ground}(i)} \mathsf{PR}[b_i \notin S_1^* \text{ and } b_i \notin S_1^* \text{ and } \ldots \text{ and } b_i \notin S_{\lceil \log m \rceil + 2}^*] \\
&= 1 - \sum_{b_i \in \mathsf{ground}(i)} \prod_{j=1}^{\lceil \log m \rceil + 2} \mathsf{PR}[b_i \notin S_j^*] \\
&\geq 1 - \sum_{b_i \in \mathsf{ground}(i)} \prod_{j=1}^{\lceil \log m \rceil + 2} 1/2 \\
&= 1 - \sum_{b_i \in \mathsf{ground}(i)} \prod_{j=1}^{\lceil \log m \rceil + 2} \mathsf{PR}[b_i \notin S_j^*] \\
&\geq 1 - \sum_{b_i \in \mathsf{ground}(i)} 1/4m \\
&= 1 - 1/4 \\
&= 3/4
\end{aligned}$$

and thus $\mathbb{E}[f(\hat{S})] \geq 3/4 f(\mathsf{ground}(f))$. On the other hand, by the linearity of expectation and the fact that f is subadditive we have:

$$\begin{aligned}
\mathbb{E}[f(\hat{S})] &= \mathbb{E}[f(\bigcup S_i^*)] \\
&\leq \mathbb{E}[\sum f(S_i^*)] \\
&\leq (\lceil \log m \rceil + 2)(\sum_{S \subseteq \mathsf{ground}(f)} \alpha_S f(S))
\end{aligned}$$

Therefore $\sum_{S\subseteq \mathsf{ground}(f)} \alpha_S f(S) \geq 3/4 f(\mathsf{ground}(f))/(\lceil \log m \rceil + 2)$, which means

$$\sum_{S\subseteq \mathsf{ground}(f)} \alpha_S f(S) \geq f(\mathsf{ground}(f))/(2\lceil \log m \rceil)$$

for big enough m. This shows the optimal solution of LP 2.73 is lower bounded by $f(\mathsf{ground}(f))/(2\lceil \log m \rceil)$ and the proof is complete. □

In what follows, based on Lemma 2.8.1, we provide a reduction from subadditive agents to XOS agents. An immediate corollary of Lemma 2.8.1 is the following:

Corollary 2.8.2 (of Lemma 2.8.1) *For any subadditive function f and integer number n, there exists an XOS function g such that*

$$g(S) \leq f(S) \qquad \forall S \subseteq \mathsf{ground}(f)$$

and

$$\mathsf{MMS}_g^n \geq \mathsf{MMS}_f^n / 2\lceil \log n \rceil.$$

Proof. By definition, we can divide the items into n disjoint sets such that the value of f for every set is at least MMS_f^n. Now, based on Lemma 2.8.1, we approximate f for each set with an additive function g_i wile losing a factor of at most $\lceil 2|\log \mathsf{ground}(f)|\rceil$ and finally we set $g = \max g_i$. Based on Lemma 2.8.1, both conditions of this lemma are satisfied by g. □

Based on Theorem 2.7.4 and Lemma 2.8.2 one can show that a $1/10\lceil \log m \rceil$-MMS allocation is always possible for subadditive agents.

Theorem 2.8.3 *The fair allocation problem with subadditive agents admits a* $1/10\lceil\log m\rceil$*-*MMS *allocation.*

Chapter 3: Almost Envy-free Allocation of Indivisible Goods

3.1 Introduction

Fair division is a fundamental and interdisciplinary problem that has been extensively studied in economics, mathematics, political science, and computer science [52, 53, 54, 55, 56, 57, 58, 4, 59, 19, 60, 5, 61]. Generally, the goal is to find an allocation of a resource to n agents, which is agreeable to all the agents according to their preferences. The first formal treatment of this problem was in 1948 by Steinhaus [62]. Following this work, a vast literature has been developed and several notions for measuring fairness have been suggested [62, 17, 1, 4, 61]. One of the most prominent and well-established fairness notions, introduced by Foley [17], is envy-freeness, which requires that each agent prefers his share over that of any other agent.

Traditionally, envy-freeness has been studied for both divisible and indivisible resources. When the resource is a single heterogeneous divisible item (i.e, can be fractionally allocated), envy-freeness admits strong theoretical guarantees. For example, it is shown that allocations exist that allocate the entire resource, and are both envy-free and Pareto efficient[1] and allocate each agent a contiguous piece of the resource [63]. Apart

[1]An allocation is Pareto efficient if it is not possible to reallocate the resources such that at least one agent is better off without making any other person worse off.

from mere existence, there are algorithms that find an envy-free allocation for arbitrary number of agents [64, 18, 65]. However, beyond divisibility, when dealing with a set of indivisible goods, envy-freeness is too strong to be attained; for example, for two agents and a single indivisible good, the agent that receives no good envies another party. Therefore, several relaxations of envy-freeness are introduced for the case of indivisible items [4, 1, 61]. One of these relaxations, suggested by Budish [1], is *envy-freeness up to one good* (EF1)[2]. An allocation of indivisible goods is EF1 if any possible envy of an agent for the share of another agent can be resolved by removing some good from the envied share. In contrast to envy-freeness, EF1 allocation always exists. Indeed, a simple round-robin algorithm always guarantees EF1 for additive valuations, and a standard envy-graph based allocation guarantees EF1 for more general (sub-additive) valuations. Besides, it is shown that any Nash welfare maximizing allocation (allocation that maximizes the product of the agents' utilities) is both Pareto efficient and EF1.

Recently, Caragiannis *et al.* [61] suggested another intriguing relaxation of envy-freeness, namely *envy-free up to any good* (EFX), which attracted a lot of attention. An allocation said to be EFX, if no agent envies another agent after the removal of any item from the other agent's bundle. Theoretically, this notion is strictly stronger than EF1 and is strictly weaker than EF. In contrast to EF1, questions related to EFX notion is relatively unexplored. As an example, despite significant effort [61], the existence of such allocations is still unknown, even for the case of three agents and additive valuations. Furthermore, unlike EF1, Nash social welfare maximizing allocations are not necessarily

[2]It is worth to mention that before the work of Budish [1] EF1 was implicitly addressed by Lipton et. al [4].

EFX [61].

Given this impenetrability of EFX, a growing strand of research started considering its relaxations. For example, Plaut and Roughgarden [58], consider an approximate version of EFX[3] and provide a $1/2$ approximation solution for agents with sub-additive valuation functions. For additive valuations, this factor is recently improved to 0.618 by Amanatidis *et al.* [5]. Another interesting relaxation is EFX-*with-charity*. Such allocations donate a bundle of items to charity and guarantee EFX for the rest of the items. The less valuable the donated items are, the more desirable the allocation is. Caragiannis *et al.*[66] show that there always exists an EFX-with-charity allocation where every agent receives half the value of his bundle in the optimal Nash social welfare allocation. Recently, Chaudhury *et al.* [60] have proposed an EFX-with-charity allocation such that no agent values the donated items more than his bundle and the number of donated items is less than the number of agents.

Considering the huge discrepancy between EFX and EF1, in this paper we wish to find a middle ground to bridge this gap. We therefore suggest another fairness criterion, namely *envy-freeness up to a random item* or EFR, which is weaker than EFX, yet stronger than EF1. For this notion, we provide a polynomial time 0.72-approximation algorithm, i.e., an algorithm that constructs 0.72-EFR allocations in polynomial time. Our allocation method is based on a special type of matching, namely Nash Social Welfare Matching. In Section 3.1.1, we briefly discuss our techniques to obtain these results.

[3]An allocation is α- approximate EFX, if for every pair of agents i and j, agent i believes that the share allocated to him is worth at least α fraction of the share allocated to agent j, after removal of agent j's least valued item (according to agent i's preference).

3.1.1 Our Results and Techniques

Envy-freeness up to a random item. We suggest a new fairness notion, namely *evny-free up to a random good* (EFR). Roughly speaking, in an EFR allocation, no agent i envies another agent j (in expectation), if we remove a random good from the bundle of agent j. In other words, the expected value of agent i for the bundle allocated to agent j, after removing a random item from it is at most as much as the value of his own bundle. Obviously, EFR is a weaker notion than EFX, yet stronger than EF1.

The intuition behind EFR is to use randomness to reduce the *severe impact of small items*. To see what we mean by this term, consider the following scenario: suppose that the value of agent i for his share is 1000. In addition, assume that the bundle allocated to an agent j contains two items, each with value 600 to agent i. Even though the allocation is currently EFX with respect to agent i, allocating even a very small item (say, with value 0 to agent i) to agent j violates EFX condition for agent i. This is a bit strange since the last item allocated to agent j was totally worthless to agent i. However, allocating any item with value less than 300 to agent j preserves EFR condition for agent i. This property makes EFR more flexible, especially when the number of items is not too much. On the other hand, as the number of items allocated to an agent grows larger, we expect EFX and EFR to be more and more aligned.

Similar to EFX, we provide a counter example which shows that a Nash Social Welfare allocation is not necessarily EFR. This separates EFR and EF1 given the fact that a Nash Social welfare allocation is always EF1[61]. It is worth mentioning that Caragiannis et al. [61] presented an example to show that Nash Social welfare allocation is not necessarily EFX. However, their example is still EFR. The difference between these two examples can be seen as an evidence for the distinction between EFR and EFX.

As noted, the best approximation guarantee for EFX is 0.61 by Amanatidis et al. [5]. Since every EFX allocation is also EFR, this result also provides a 0.61-approximation algorithm for EFR. In this paper, we improve this ratio to 0.72.

theoremefrthm There exists an algorithm that finds a 0.72-EFR allocation. In addition, such an allocation can be found in polynomial time.

In order to prove Theorem 3.1.1, we propose a three-step algorithm that finds a 0.72-EFR allocation in polynomial time. Roughly speaking, in the first two steps, we allocate valuable (i.e., large) items while preserving the 0.72-EFR property. Next, we use an envy-cycle based procedure to allocate the rest of the items. Figure 3.1 shows a flowchart of our method.

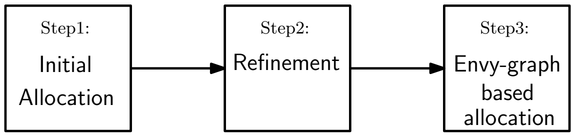


Figure 3.1: Flowchart of the 0.72-EFR allocation algorithm

The first challenge to address is the method by which we must allocate large items in the first step. Interestingly, we introduce a special type of matching allocation with intriguing properties which makes it ideal for our algorithm. We call such an allocation a *Nash Social Welfare Matching*.

Nash Social Welfare Matching. In the first step of the algorithm, we allocate one item to each agent such that the product of the utilities of the agents is maximized. The interesting fact about this allocation is that, not only does this allocation allocates large items, but it also provides very useful information about the value of the rest of the items. In Section 3.2 we broadly discuss such allocations and their properties. However, to shed light on their usefulness, assume that after a Nash Social Welfare Matching, agent i envies agent j with a ratio $\alpha > 1$, meaning that he thinks the value of the good allocated to agent j is α times more than the value of his item. In that case, we can immediately conclude that the item allocated to agent j is α times more valuable to him (agent j) than any remaining item; otherwise, we could improve the utility product by allocating the most valuable remaining item to agent j and giving his former item to agent i (and of course, freeing agent i's former item). In addition, we can express the same proposition for the value of the item allocated to agent i for agent j: the value of this item for agent j is at most $1/\alpha$ of the item allocated to agent j. The above statement can be generalized to the arguments that include more than two agents. With this aim, we introduce several new concepts, including *envy-ratio graph* (a complete weighted graph that represents the envy-ratios between agents), *improving cycles*, and *envy-rank*.

It is worth mentioning that the main challenge in many fair allocation problems for different fairness criteria (e.g., MMS, EFX) is allocating valuable items. The structure of such matchings makes them ideal for allocating these items. We strongly believe that using Nash Social Welfare matching is not only useful for our algorithm, but can also be seen as a strong tool in the way of finding fair allocations related to the other fairness notions, especially maximin-share. In Section 3.4 we show how to use NSW mathcing to obtain the same approximation ratio as the state-of-the-art $(\phi - 1)$ [67] for EFX.

3.1.2 Related work

Fair allocation of a divisible resource (konwn as *cake cutting*) was first introduced by Steinhaus [62], and since then has been the subject of intensive studies. We refer the reader to [68] and [69] for an overview on different fairness notions and their related results. Proportionality and Envy-freeness are among the most important notions for cake cutting. As mentioned, the literature of cake cutting admits strong positive results for these two notions (see [62] for more details).

Since neither EF nor proportionality or any approximation of these notions can be guaranteed for indivisible items, several relaxations are introduced for these two notions. These relaxations include EF1 as well as EFX for envy-freeness and maximin-share [1] as well as maximin fairness [70] for proportionality. In addition, the weighted version of these notions (where the agents have different entitlements) are also considered. Nash Social Welfare (NSW) is another important notion in allocation of indivisible goods which is somewhat a trade off between fairness and optimality.

Apart from the results mentioned in the previous section for EFX and EF1, there are other studies related to these notions. For example, Barman *et al.* [71] proposed a pseudo-polynomial time algorithm that finds an EF1 and pareto efficient allocation. Furthermore, they show that any EF1 and pareto efficient allocation approximates Nash Social Welfare with a factor of 1.45. It is worth mentioning that the problem of whether a (strongly) polynomial time algorithm can guarantee both EF1 and Pareto optimality is still open.

Maximin-share is one of the most well-studied notions in the recent years. In a pioneering study, Kurokawa *et al.* [57] provided an approximation algorithm with the factor of $2/3$ for maximin-share, and Amanatidis *et al.* [72] show that their method can be implemented in polynomial time to obtain a $2/3-\epsilon$ approximation guarantee. Barman *et al.* [19] show that a simple round robin algorihtm can guarantee the approximation of $2/3$ for additive, and $1/10$ for submodular valuations. This is later improved to $3/4$ by Ghodsi *et al.* [54] and Garg *et al.* [73]. Ghodsi *et al.* also provided approximation guarantees for submodular $(1/3)$, XOS $(1/5)$ and subadditive $(1/\log n)$ valuations.

In addition, several notions are ramified from maximin-share. For example weighted maximin-share (WMMS) [52] and pairwise maximin-share (PMMS) [61]. Several studies consider the relation between these notions and seek to find an allocation that guarantees a number of these notions simultaneously. For example, Amanatidis *et al.* [67] investigate the connections between EF1 and EFX allocations with two other famous fairness notions, namely maximin share fairness (MMS) and pairwise maximin share fairness (PMMS). They show that any EF1 allocation is also a $1/n$-MMS and a $1/2$-PMMS allocation. They also proved that any EFX allocation is a $4/7$-MMS and a $2/3$-PMMS allocation.

3.2 Preliminaries and Basic Observations

Fair allocation problem. An instance of fair allocation problem is consisted a set of n agents, a set $\mathcal{M}$ of m goods, and a valuation profile $V = \{V_1, V_2, \ldots, V_n\}$. Each v_i is a function of the form $2^{\mathcal{M}} \rightarrow \mathbb{R}_{\geq 0}$ which specifies the preferences of agent $i \in [n]$ over the goods. Throughout the paper, we assume that a valuation function v_i satisfies the following conditions.

- **Normalization**: $V_i(\varnothing) = 0$.
- **Monotonicity**: $V_i(S) \leq V_i(T)$ whenever $S \subseteq T$.
- **Additivity**: $V_i(S) = \sum_{b \in S} V_i(\{b\})$.

An allocation of a set S of goods is an n-partition $\mathcal{A} = \langle \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n \rangle$ of S, where $\mathcal{A}_i$ is the bundle allocated to agent i. Allocation is complete, if $S = \mathcal{M}$ and is partial otherwise. Since we are interested in the allocations that allocate the whole set of items, the final allocation must be complete.

Fairness critera. Given an instance of fair division problem and an allocation $\mathcal{A}$, an agent i envies another agent j, if he strictly prefers $\mathcal{A}_j$ over his bundle $\mathcal{A}_i$. An allocation is then said to be *envy-free* (EF), if no agent envies another, i.e., for every pair $i, j \in [n]$ of agents we have $V_i(\mathcal{A}_i) \geq V_i(\mathcal{A}_j)$. As mentioned, envy-freeness is too strong to be guaranteed in an allocation of indivisible items. Therefore, two relaxations of this notion are introduced, namely *envy-free up to one good* (EF1) and *envy-free up to any good* (EFX).

Definition 3.2.1 *An allocation $\mathcal{A}$ is called*

- *envy-free up to one good (*EF1*) if for all i, j we have $V_i(\mathcal{A}_i) \geq \min_{b \in \mathcal{A}_j} V_i(\mathcal{A}_j \setminus \{b\})$,*
- *envy-free up to any good (*EFX*) if for all i, j we have $V_i(\mathcal{A}_i) \geq \max_{b \in \mathcal{A}_j} V_i(\mathcal{A}_j \setminus \{b\})$.*

Even though these two notions seem to be somewhat related, there is a huge discrepancy between the current results obtained for them. It is shown that even for instances with general valuations, an EF1 allocation always exists, and can be computed in polynomial time [4]. In contrast, whether or not an EFX allocation exists is still open, even for additive valuations and 3 agents.

In this paper, we introduce another relaxation of envy-freeness, namely *envy-free up to a random good*. Let D_j be a uniform distribution over the items of $\mathcal{A}_j$ that selects each item with probability $1/|\mathcal{A}_j|$.

Definition 3.2.2 *Allocation $\mathcal{A}$ is envy-free up to a random good (*EFR*) if for all i, j we have*

$$V_i(\mathcal{A}_i) \geq \mathop{\mathbb{E}}_{b \sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j \setminus \{b\})\right].$$

Clearly, EFR lies in between EFX and EF1: EFX is a stronger notion that EFR, and EFR is stronger than EF1. In Example 1, we show one structural difference between EF1 and EFR: in contrast to EF1, EFR is not implied by an allocation that maximizes Nash social welfare.

	1	2	3	4	5
V_1	3	3	1	1	1
V_2	5	5	1	4	3

Figure 3.2: Agents' valuations over items

Example 1 *Consider an instance of the fair allocation problem with 5 items, and 2 agents with the valuations represented in Figure 3.2. The unique allocation that maximizes the* NSW *allocates the first 3 items to the first agent, and the other 2 to the second agent. Let*

$\mathcal{A}$ be this allocation. Since there are 3 items in the first agent's bundle, we have

$$\mathop{\mathbb{E}}_{b\sim \mathsf{D}_1}\Big[V_2(\mathcal{A}_1\setminus\{b\})\Big] = \frac{1}{3}\cdot\big(V_2(\mathcal{A}_1\setminus\{1\})+V_2(\mathcal{A}_1\setminus\{2\})+V_2(\mathcal{A}_1\setminus\{3\})\big)$$
$$= \frac{22}{3} \geq V_2(\mathcal{A}_2) = 7,$$

and hence, this allocation is not EFR.

Finally, approximate versions of EFX and EFR are defined as follows.

Definition 3.2.3 *For a constant $c \leq 1$, an allocation $\mathcal{A}$ is called*

- *c-approximate envy-free up to any good (c-*EFX*), if for all i, j we have*

$$V_i(\mathcal{A}_i) \geq c \cdot \max_{b\in\mathcal{A}_j} V_i(\mathcal{A}_j\setminus\{b\})\,.$$

- *c-approximate envy-free up to a random good (c-*EFR*) if for all i, j we have*

$$V_i(\mathcal{A}_i) \geq c \cdot \mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\Big[V_i(\mathcal{A}_j\setminus\{b\})\Big]\,.$$

Note that Example 1 also shows that the maximum NSW *allocation does not guarantee better than $\frac{21}{22}$ approximation of* EFR.

Envy-ratio Graph. Envy-ratio graph is in fact a generalization of envy-graph introduced by Lipton et al. [4]. Suppose that at some stage of our algorithm we have a partial allocation $\mathcal{A}$. We define a graph called *envy-ratio graph* to be a complete weighted digraph with the following construction: each vertex corresponds to an agent, and for each ordered pair (i, j), there is a directed edge from vertex i to vertex j with weight $w_{i,j} = V_i(\mathcal{A}_j)/V_i(\mathcal{A}_i)$.

Assuming each agent has a non-zero value for each good, for every i, j we have $w_{i,j} \in [0, \infty)$. Note that $w_{i,j} \leq 1$ implies that agent i does not envy agent j, whereas $w_{i,j} > 1$ indicates agent i envies agent j. The higher the value of $w_{i,j}$ is, the more envious agent i is to the bundle of agent j. Indeed, the well-known envy-graph is a subgraph of envy-ratio graph containing only the edges with $w_{i,j} > 1$.

Nash Social Welfare (NSW) Mathcing. Nash social welfare, originally proposed by Nash [59], is defined to be the geometric mean of agents' valuations. Allocation that maximizes Nash social welfare is known to have desirable properties. For example, such allocations are proved to be EF1 and pareto optimal. Roughly, Nash social welfare maximizing allocations can be seen as a trade-off between social welfare and fairness.

In the first step of the algorithm, we allocate one item to each agent such that the Nash social welfare of the agents is maximized. More formally, define Nash Social Welfare matching of $[m]$ to be a partial allocation $\mathcal{A} = \langle \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n \rangle$, such that $\Pi_i V_i(\mathcal{A}_i)$ is maximized and where for every i we have $|\mathcal{A}_i| = 1$.

Similar to Nash social welfare allocations, Nash social welfare matchings exhibit beautiful properties which greatly help us in designing our algorithm. One simple property of such allocations is shown in Observation 3.2.5. Before we state Observation 3.2.5, we need to define concepts of *improving* and *strictly improving* cycles.

Definition 3.2.4 *Let $c = i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_k \rightarrow i_1$ be a cycle in the envy-ratio graph.*

Then, c is an improving cycle, if

$$w_{i_1,i_2} \times w_{i_2,i_3} \times \ldots \times w_{i_{k-1},i_k} \times w_{i_k,i_1} > 1\,.$$

Furthermore, we say a cycle c is strictly improving cycle, if c is an improving cycle and for every $(i \rightarrow j) \in c$ we have $w_{i,j} > 1$.

We note that strictly improving cycle is an essential concept in all envy-cycle elimination methods [4, 19, 60, 5]. These methods typically rotate the shares over strictly improving cycles to enhance social welfare. However, to the best of our knowledge, no previous work made use of improving cycles.

Observation 3.2.5 *Suppose that we allocate one item to each agent using Nash social welfare matching. Then, the envy-ratio graph admits no improving cycle.*

Proof. Assume $c = i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_k \rightarrow i_1$ is an improving cycle. Then, it is easy to see that rotating the goods over this cycle (i.e., reallocating $\mathcal{A}_{i_j}$ to agent i_{j-1} for every $1 < j \leq k$, and reallocating $\mathcal{A}_{i_1}$ to agent i_k) yields a matching with a higher Nash social welfare. $\square$

A particularly useful case of Observation 3.2.5 is for the cycles of length 2, which we state in Corollary 3.2.6.

Corollary 3.2.6 (of Observation 3.2.5) *Suppose that for two agents i, j we have $V_i(\mathcal{A}_j) \geq r \cdot V_i(\mathcal{A}_i)$, where $r \geq 1$. Then, we have $V_j(\mathcal{A}_i) \leq V_j(\mathcal{A}_j)/r$.*

Definition 3.2.7 *Suppose that we allocate one item to each agent using Nash social welfare matching. We define the envy-rank of an agent i, denoted by r_i as*

$$r_i = \max_{j_0, j_1, \dots, j_k} \prod_{z=1}^{k} w_{j_z, j_{z-1}} ,$$

where $j_0 = i$. Roughly speaking, let p be a path leading to vertex i such that the product of the weights of the edges in p is maximum. Then, envy-rank of agent i equals to the product of the weights of the edges in p. Note that by Observation 3.2.5 we can assume w.l.o.g that p is a simple path (i.e., p includes no duplicate vertices).

Observation 3.2.8 *p is a simple path.*

Proof. Assume p is not simple and let c be a cycle in p. By Observation 3.2.5 we know that c can not be improving. Therefore, the product of the weight of the edges of $p \setminus c$ is as large as p. $\square$

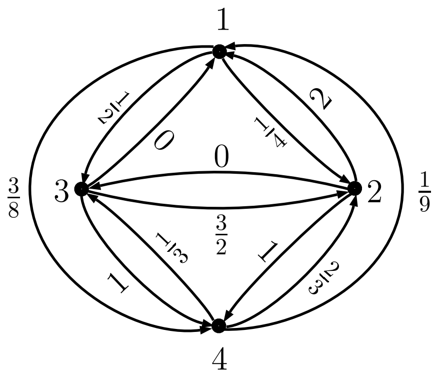
To get a better understanding of these definitions take a look at Example 2.

Example 2 *Consider an instance of the fair allocation problem with 4 items, 4 agents, and a valuation profile $V = \{V_1, V_2, V_3, V_4\}$ represented in Figure 3.3a. Let $\mathcal{A}$ be the allocation that allocates item i to agent i. The envy-ratio graph and the envy graph of $\mathcal{A}$ are shown in Figure 3.3b and Figure 3.3c respectively. This allocation is not envy-free, however, it is both* EFX *and* EFR *since each agent receives only one item.*

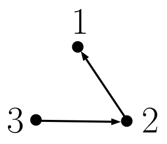
As we mentioned before, the envy-rank of an agent can be seen as the product of the weights of the edges in a path leading to that agent. For instance, consider the agent 1. *The envy-rank of this agent is* 3 *which is the product of the weights of the edges in*

	1	2	3	4
V_1	8	2	4	3
V_2	4	2	0	2
V_3	0	3	2	2
V_4	1	6	3	9

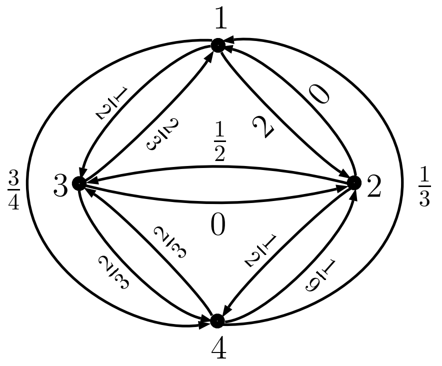
(a) Agents' valuations over items



(b) Envy-ratio graph



(c) Envy graph



(d) Envy-ratio graph after eliminating a cycle.

Figure 3.3: An example to illustrate envy-ratio graph.

the path $3 \to 2 \to 1$. *Also consider the cycle* $1 \to 3 \to 2 \to 1$. *This cycle is an improving cycle. Therefore, the allocation* $\mathcal{A}$ *is not a* NSW *matching. The allocation can be improved by moving the items alongside this cycle which leads to a new allocation* $\mathcal{A}' = \langle \{3\}, \{1\}, \{2\}, \{4\} \rangle$. *The envy-ratio graph of* $\mathcal{A}'$ *can be seen in Figure 3.3d.*

We finish our discussion in this section by mentioning some properties of envy-rank values.

Observation 3.2.9 *Suppose that allocation* $\mathcal{A}$ *allocates one item to each agent using a*

Nash social welfare matching. Then for every pair of agents i and j, we have

$$\frac{V_i(\mathcal{A}_j)}{V_i(\mathcal{A}_i)} \leq \min\left\{r_j, \frac{r_j}{r_i}\right\}.$$

Proof. In the envy-ratio graph, the weight of the directed edge from i to j is $w_{i,j} = \frac{V_i(\mathcal{A}_j)}{V_i(\mathcal{A}_i)}$. Recall that r_j is the maximum product of the weights of the edges in a path leading to j. Since the edge from i to j is also a path leading to vertex j, we have $\frac{V_i(\mathcal{A}_j)}{V_i(\mathcal{A}_i)} = w_{i,j} \leq r_j$. Now consider a path p leading to i with the maximum product of the weights of the edges. Based on the definition of envy-rank, the product of the weights of the edges in p is r_i. We can use the edge from i to j to extend this path. This new path leads to j, and its product of the weights of the edges is $r_i \cdot w_{i,j}$. Therefore, we can say $r_j \geq r_i \cdot w_{i,j}$. Hence,

$$\frac{V_i(\mathcal{A}_j)}{V_i(\mathcal{A}_i)} = w_{i,j} \leq \frac{r_j}{r_i}.$$

□

In addition to Observation 3.2.5, Nash social welfare matchings admit another important and elegant property, which we state in Observation 3.2.10. This observation provides upper bounds on the value of remaining goods and can be of independent interest for various fair allocation problems.

Observation 3.2.10 *Suppose that we allocate one item to each agent using a Nash social welfare matching. Then, for each agent i and any unallocated item b we have*

$$V_i(b) \leq \min\left\{V_i(\mathcal{A}_i), \frac{V_i(\mathcal{A}_i)}{r_i}\right\}.$$

Proof. First, for any agent i and any remaining good b, we have $V_i(b) \leq V_i(\mathcal{A}_i)$, if not NSW can be increased by giving b to i. On the other hand, consider a path leading to i with the maximum product of the weights of the edges. By moving items along this path and giving b to agent i, the NSW will be multiplied by a factor of $r_i \cdot \frac{V_i(b)}{V_i(\mathcal{A}_i)}$. Since $\mathcal{A}$ is the allocation that maximizes NSW, we have $r_i \cdot \frac{V_i(b)}{V_i(\mathcal{A}_i)} \leq 1$, and hence $V_i(b) \leq \frac{V_i(\mathcal{A}_i)}{r_i}$. □

3.3 An approximate EFR Allocation

In this section, we present our algorithm for finding a 0.72-EFR allocation. Our algorithm is divided into 3 steps, namely NSW matching, allocation refinement, and envy-graph based allocation. In the first step, we allocate each agent one item using a Nash social welfare matching and accordingly divide the agents into three groups based on their envy-rank. Next, in the second step we allocate a set of goods to the agents in each group, and finally in the third step we allocate the rest of the items using the classic envy-cycle elimination method. The outline of our algorithm is represented in Algorithm 7.

3.3.1 Step 1.

In the first step, we allocate one item to each agent using a NSW matching. We first show that this allocation can be found in polynomial time.

Observation 3.3.1 NSW *matching can be found in polynomial time.*

Proof. Let $G = (U_1, U_2)$ be a bipartite graph that has a vertex for each agent in U_1 and has a vertex for every item in U_2. For every agent i and every item b we add an undirected

Algorithm 7: The outline of the 0.72-EFR algorithm.

```
// Step 1
(𝒜, ℳ′) = NSWMatchingAllocation(V, ℳ);
(G_1, G_2, G_3) = EnvyRankPartition(𝒜);
// Step 2
𝒪 = TopologicalOrderedAgents(𝒜);
foreach $i \in \mathsf{G}_3$ ordered by $\mathcal{O}$ do
    $(\mathcal{A}, \mathcal{M}')$ = ExtendAllocation($\mathcal{A}$, $\mathcal{M}'$, $i$);
end
foreach $i \in \mathsf{G}_3$ ordered by $\mathcal{O}$ do
    $(\mathcal{A}, \mathcal{M}')$ = ExtendAllocation($\mathcal{A}$, $\mathcal{M}'$, $i$);
end
foreach $i \in \mathsf{G}_2$ ordered by $\mathcal{O}$ do
    $(\mathcal{A}, \mathcal{M}')$ = ExtendAllocation($\mathcal{A}$, $\mathcal{M}'$, $i$);
end
// Step 3
while $\mathcal{M}' \neq \varnothing$ do
    Eliminate all directed cycles in the envy-graph;
    Let $s$ be an arbitrary source in the envy graph;
    $(\mathcal{A}, \mathcal{M}')$ = ExtendAllocation($\mathcal{A}$, $\mathcal{M}'$, $s$);
end
return $\mathcal{A}$;
```

edge with the weight of $\log V_i(\{b\})$ between their corresponding vertices. By finding a maximum weighted matching in this graph, we get an allocation $\mathcal{A}$ such that every agent has at most one allocated item. Also, this allocation maximizes $\sum_{i=1}^{n} \log V_i(\mathcal{A}_i)$. Therefore, this allocation also maximizes $\prod_{i=1}^{n} V_i(\mathcal{A}_i)$. Hence, $\mathcal{A}$ allocates at most one item to every agent and maximizes Nash social welfare. $\square$

Let $\mathcal{A}$ be NSW matching and fix a parameter $\varphi = 8/3$. Based on the envy-rank of the agents, we divide them into 3 groups $\mathsf{G}_1, \mathsf{G}_2$, and G_3 as follows.

- Agent i belongs to G_1 if $r_i > \varphi$.
- Agent i belongs to G_2 if $2 < r_i \leq \varphi$.
- Agent i belongs to G_3 if $r_i \leq 2$.

Note that by Observation 3.2.10, we know that for every remaining item b the following properties hold.

- (Property 1): For every agent $i \in \mathsf{G}_1$ we have $V_i(b) < V_i(\mathcal{A}_i)/\varphi$.
- (Property 2): For every agent $i \in \mathsf{G}_2$ we have $V_i(b) < V_i(\mathcal{A}_i)/2$.

Intuitively, if each remaining item is worth less than $V_i(\mathcal{A}_i)/\varphi$ to every agent i, then we can guarantee the approximation factor of $1/(1 + 1/\varphi)$ in the third step. This property holds for the agents in G_1; however, this is not the case for agents in G_2 and G_3. In the second step, we seek to allocate a set of items to the agents in G_2 and G_3 so that the same property holds for these agents. Note that alongside this property, the final partial allocation after the second step must be fair (i.e., 0.72-EFR).

3.3.2 Step 2.

In the second step, we allocate one item to each agent in G_2 and two items to each agent in G_3. Algorithm 7 shows the method by which we allocate these items to the agents in G_2 and G_3. Let $\mathcal{O}$ be a topological ordering of the agents with respect to the envy-graph. We order the agents in G_3 according to $\mathcal{O}$ and ask them one by one to pick their most valuable remaining good. We then again ask agents in G_3 to pick one more item according to the same topological ordering $\mathcal{O}$. Afterwards, we order the agents in G_2 according to $\mathcal{O}$ and ask them one by one to add the most desirable remaining item to their bundles.

After this step the bundle of every agent in G_3 contains three items. For an agent $i \in \mathsf{G}_3$ we use $\{b_i, b'_i, b''_i\}$ to denote the items allocated to this agent where b'_i and b''_i are the items allocated in Step 2 and b'_i has been allocated before b''_i. Also, the bundle of every agent in G_2 contains two items. Similarly, for an agent $i \in \mathsf{G}_2$ we use $\{b_i, b'_i\}$ to denote the allocated items of this agent where b'_i is the item received in Step 2. We also use $\{b_i\}$ to denote the only item received by an agent $i \in \mathsf{G}_1$.

We begin our analysis by showing the following claim.

Claim 3.3.2 *For an allocation $\mathcal{A}$ and agents i and j, we have*

$$\mathop{\mathbb{E}}_{b \sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j \setminus \{b\})\right] = \frac{|\mathcal{A}_j| - 1}{|\mathcal{A}_j|} \cdot V_i(\mathcal{A}_j)\,.$$

Proof. Distribution D_j selects each item in $\mathcal{A}_j$ with the probability of $1/|\mathcal{A}_j|$. Therefore,

$$\begin{aligned}\mathop{\mathbb{E}}_{b\sim\mathsf{D}_j}\left[V_i(\mathcal{A}_j\setminus\{b\})\right] &= \frac{1}{|\mathcal{A}_j|}\cdot\sum_{b\in\mathcal{A}_j} V_i(\mathcal{A}_j\setminus\{b\}) \\ &= \frac{1}{|\mathcal{A}_j|}\cdot\sum_{b\in\mathcal{A}_j}\sum_{b'\in\mathcal{A}_j\setminus\{b\}} V_i(\{b'\})\,. \qquad \text{By Additivity assumption.}\end{aligned}$$

Each item in $\mathcal{A}_j$ appears $|\mathcal{A}_j|-1$ times in the above summation. Therefore,

$$\begin{aligned}\mathop{\mathbb{E}}_{b\sim\mathsf{D}_j}\left[V_i(\mathcal{A}_j\setminus\{b\})\right] &= \frac{1}{|\mathcal{A}_j|}\cdot\sum_{b\in\mathcal{A}_j}\sum_{b'\in\mathcal{A}_j\setminus\{b\}} V_i(\{b'\}) \\ &= \frac{|\mathcal{A}_j|-1}{|\mathcal{A}_j|}\cdot\sum_{b\in\mathcal{A}_j} V_i(\{b\}) = \frac{|\mathcal{A}_j|-1}{|\mathcal{A}_j|}\cdot V_i(\mathcal{A}_j)\,.\end{aligned}$$

□ We now show that at the end of Step 2 the following conditions hold.

- The current allocation is EFR with respect to the agents in G_1.
- The current allocation is $(3/4)$-EFR with respect to the agents in G_2 and G_3.

Claim 3.3.3 *By the end of Step 2, the allocation is* EFR *with respect to the agents in* G_1.

Proof. Let i be an agent in G_1, we show that for every other agent j we have

$$V_i(\mathcal{A}_i) \geq \mathop{\mathbb{E}}_{b\sim\mathsf{D}_j}\left[V_i(\mathcal{A}_j\setminus\{b\})\right],$$

so the allocation is EFR from the agent i's perspective. If $j\in\mathsf{G}_1$, then we have $|\mathcal{A}_j|=1$ and the claim clearly holds. Consider an agent $j\in\mathsf{G}_2$. At the end of Step 2, agent j has two allocated items. By Observation 3.2.10 the valuation of the item b'_j for agent i is bounded by $V_i(\mathcal{A}_i)/r_i = V_i(b_i)/r_i$. Therefore,

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j \setminus \{b\})\right] &= \frac{V_i(b_j) + V_i(b'_j)}{2} && \text{Claim 3.3.2.}\\
&\le \frac{V_i(b_j) + V_i(b_i)/r_i}{2} && \text{Observation 3.2.10.}\\
&\le \frac{V_i(b_i) + V_i(b_i)/r_i}{2} && \text{Observation 3.2.9.}\\
&= \frac{1 + 1/r_i}{2} \cdot V_i(b_i)\,.
\end{aligned}$$

Since agent i is in G_1, we have $r_i \ge \varphi$. It follows that

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j \setminus \{b\})\right] &\le \frac{1 + 1/r_i}{2} \cdot V_i(b_i)\\
&\le \frac{1 + 1/\varphi}{2} \cdot V_i(b_i)\\
&= \frac{11}{16} \cdot V_i(b_i)\\
&\le V_i(b_i) = V_i(\mathcal{A}_i)\,.
\end{aligned}$$

The only remaining case is when agent j is in G_3. Since agent j has exactly three allocated items we have,

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j\setminus\{b\})\right] &= \frac{2}{3}\cdot\big(V_i(b_j)+V_i(b'_j)+V_i(b''_j)\big) && \text{Claim 3.3.2.}\\
&\le \frac{2}{3}\cdot\big(V_i(b_j)+2V_i(b_i)/r_i\big) && \text{Observation 3.2.10.}\\
&\le \frac{2}{3}\cdot\big(r_j\cdot V_i(b_i)/r_i+2V_i(b_i)/r_i\big) && \text{Observation 3.2.9.}\\
&= \frac{2r_j/r_i+4/r_i}{3}\cdot V_i(b_i)\,.
\end{aligned}$$

Recall that agents i and j are in G_1 and G_3 respectively. Therefore, $r_i \ge \varphi$ and $r_j \le 2$. We then have

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j\setminus\{b\})\right] &\le \frac{2r_j/r_i+4/r_i}{3}\cdot V_i(b_i)\\
&\le \frac{8/\varphi}{3}\cdot V_i(b_i)\\
&= V_i(b_i)=V_i(\mathcal{A}_i)\,.
\end{aligned}$$

Therefore the allocation is EFR. □

Claim 3.3.4 *By the end of step 2, the allocation is* $(3/4)$-EFR *with respect to the agents in* G_2.

Proof. Let i be an agent in G_2, we show that for every other agent j, we have

$$V_i(\mathcal{A}_i)\ge 3/4\cdot \mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j\setminus\{b\})\right],$$

therefore the allocation is $(3/4)$-EFR. For an agent j in G_1, only one item is allocated to this agent and the claim clearly holds.

Consider an agent j in G_2. Recall that $\mathcal{A}_j = \{b_j, b'_j\}$ is the bundle of this agent. We first consider the case that $V_i(b_i) < V_i(b_j)$. In this case the position of agent i in $\mathcal{O}$ is before agent j. Therefore, agent i receives his second good before agent j, and we have

$$V_i(b'_i) \geq V_i(b'_j)\,. \tag{3.1}$$

Moreover, by Observation 3.2.9 we have

$$\frac{V_i(b_j)}{V_i(b_i)} \leq \frac{r_j}{r_i} < \frac{\varphi}{2} < \varphi\,. \tag{3.2}$$

It follows that

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim\mathsf{D}_j}\left[V_i(\mathcal{A}_j\setminus\{b\})\right] &= \frac{V_i(b_j)+V_i(b'_j)}{2} && \text{Claim 3.3.2.}\\
&\leq \frac{V_i(b_j)+V_i(b'_i)}{2} && \text{By (3.1).}\\
&< \frac{\varphi\cdot V_i(b_i)+V_i(b'_i)}{2} && \text{By (3.2).}\\
&\leq \frac{\varphi}{2}\cdot\big(V_i(b_i)+V_i(b'_i)\big)\\
&= \frac{3}{4}\cdot V_i(\mathcal{A}_i)\,.
\end{aligned}$$

Therefore, if $V_i(b_i) < V_i(b_j)$, the allocation is EFR. The other case is when $V_i(b_i) \geq$

$V_i(b_j)$. In that case we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\big[V_i(\mathcal{A}_j \setminus \{b\})\big] &= \frac{V_i(b_j)+V_i(b'_j)}{2} && \text{Claim 3.3.2.}\\
&\le \frac{V_i(b_j)+V_i(b_i)/r_i}{2} && \text{Observation 3.2.10.}\\
&\le \frac{V_i(b_i)+V_i(b_i)/r_i}{2} && V_i(b_i)\ge V_i(b_j).\\
&= \frac{1+1/r_i}{2}\cdot V_i(b_i)\\
&< \frac{3}{4}\cdot V_i(b_i) \le \frac{3}{4}\cdot V_i(\mathcal{A}_i)\,, && r_i > 2.
\end{aligned}$$

therefore the allocation in this case is EFR.

The only remaining case is when agent j is in G_3. Since agent j has exactly three allocated items we have,

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\big[V_i(\mathcal{A}_j \setminus \{b\})\big] &= \frac{2}{3}\cdot\big(V_i(b_j)+V_i(b'_j)+V_i(b''_j)\big) && \text{Claim 3.3.2.}\\
&\le \frac{2}{3}\cdot\big(V_i(b_j)+2V_i(b_i)/r_i\big) && \text{Observation 3.2.10.}\\
&\le \frac{2}{3}\cdot\big(r_j\cdot V_i(b_i)/r_i+2V_i(b_i)/r_i\big) && \text{Observation 3.2.9.}\\
&\le \frac{2}{3}\cdot\big(V_i(b_i)+2V_i(b_i)/r_i\big) && r_j < r_i.\\
&= \frac{2+4/r_i}{3}\cdot V_i(b_i)\\
&< \frac{4}{3}\cdot V_i(b_i) \le \frac{4}{3}\cdot V_i(\mathcal{A}_i)\,. && r_i > 2.
\end{aligned}$$

Therefore the allocation is $(3/4)$-EFR.

□

Claim 3.3.5 *By the end of step 2, the allocation is* $(3/4)$-EFR *with respect to the agents in* G_3.

Proof. Let i be an agent in G_2, we show that for every other agent j, we have

$$V_i(\mathcal{A}_i) \geq 3/4 \cdot \mathop{\mathbb{E}}_{b \sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j \setminus \{b\})\right],$$

therefore the allocation is $(3/4)$-EFR. For an agent j in G_1, the claim clearly holds since this agent has only one allocated item.

Consider an agent $j \in \mathsf{G}_2$. Agent i receives his second item before agent j in Step 2 of our algorithm. Thus, we have

$$V_i(b'_i) \geq V_i(b'_j)\,. \tag{3.3}$$

Therefore, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j\setminus\{b\})\right] &= \frac{V_i(b_j)+V_i(b'_j)}{2} && \text{Claim 3.3.2.}\\
&\le \frac{V_i(b_j)+V_i(b'_i)}{2} && \text{By (3.3).}\\
&\le \frac{r_jV_i(b_i)+V_i(b'_i)}{2} && \text{Observation 3.2.9.}\\
&\le \frac{\varphi V_i(b_i)+V_i(b'_i)}{2} && r_j\le\varphi.\\
&\le \frac{\varphi}{2}\cdot\big(V_i(b_i)+V_i(b'_i)\big)\\
&\le \frac{4}{3}\cdot V_i(\mathcal{A}_i)\,.
\end{aligned}$$

Therefore the allocation is $(3/4)$-EFR.

The remaining case is when agent j is in G_3. Consider the case that $V_i(b_i) < V_i(b_j)$. In this case the position of agent i in $\mathcal{O}$ is before agent j. Therefore, in Step 2 of our algorithm, agent i receives his second and third items before agent j, and we have the followings.

$$V_i(b'_i) \ge V_i(b'_j)\,, \tag{3.4}$$

and

$$V_i(b''_i) \ge V_i(b''_j)\,. \tag{3.5}$$

Therefore,

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j\setminus\{b\})\right] &= \frac{2}{3}\cdot\big(V_i(b_j)+V_i(b'_j)+V_i(b''_j)\big) && \text{Claim 3.3.2.}\\
&\le \frac{2}{3}\cdot\big(V_i(b_j)+V_i(b'_i)+V_i(b''_i)\big) && \text{By (3.4) and (3.5).}\\
&\le \frac{2}{3}\cdot\big(r_jV_i(b_i)+V_i(b'_i)+V_i(b''_i)\big) && \text{Observation 3.2.9.}\\
&\le \frac{2}{3}\cdot\big(2V_i(b_i)+V_i(b'_i)+V_i(b''_i)\big) && r_j\le 2.\\
&\le \frac{4}{3}\cdot\big(V_i(b_i)+V_i(b'_i)+V_i(b''_i)\big)\\
&= \frac{4}{3}\cdot V_i(\mathcal{A}_i)\,.
\end{aligned}$$

Thus, the allocation is $(3/4)$-EFR.

The other case is when $V_i(b_i)\ge V_i(b_j)$. In this case agent i receives b'_i prior to when agent j receives b''_j, and we have

$$V_i(b'_i)\ge V_i(b''_j) \tag{3.6}$$

$$\begin{aligned}
\mathop{\mathbb{E}}_{b\sim \mathsf{D}_j}\left[V_i(\mathcal{A}_j \setminus \{b\})\right] &= \frac{2}{3}\cdot\left(V_i(b_j)+V_i(b'_j)+V_i(b''_j)\right) && \text{Claim 3.3.2.}\\
&\le \frac{2}{3}\cdot\left(V_i(b_i)+V_i(b'_j)+V_i(b''_j)\right) && V_i(b_i)\ge V_i(b_j).\\
&\le \frac{2}{3}\cdot\left(V_i(b_i)+V_i(b'_j)+V_i(b'_i)\right) && \text{By (3.6).}\\
&\le \frac{2}{3}\cdot\left(V_i(b_i)+V_i(b_i)+V_i(b'_i)\right) && \text{Observation 3.2.10.}\\
&\le \frac{4}{3}\cdot\left(V_i(b_i)+V_i(b'_i)\right)\\
&\le \frac{4}{3}\cdot V_i(\mathcal{A}_i)\,,
\end{aligned}$$

and the allocation is $(3/4)$-EFR. □

3.3.3 Step 3.

In the third step, we use the envy-graph to allocate the rest of the items. We repeat the following steps until all the goods are allocated.

- Find and eliminate all the directed cycles from the envy-graph. In order to eliminate all cycles in the envy-graph, we repeatedly find a directed cycle in the envy-graph. Let $i_1 \to i_2 \to \cdots \to i_k \to i_1$ be a cycle in envy-graph. By definition, each agent i_j envies agent $i_{(j \bmod k)+1}$, i.e.,

$$v_{i_j}(\mathcal{A}_{i_j}) < v_{i_j}\left(\mathcal{A}_{i_{(j \bmod k)+1}}\right),$$

  where $\mathcal{A}$ is the current allocation. We then exchange the allocations of the agents that are in the cycle such that each agent i_j receives $\mathcal{A}_{i_{(j \bmod k)+1}}$. Note that this exchanging does not change bundles. Furthermore, the utility each agent does not decrease. Hence, if the allocation is α-EFR before the exchange, it remains α-EFR after it (*Lemma 6.1* in [58]). Also, exchanging these allocations decreases the number of edges in the envy-graph. Thus, we eventually find an allocation such that its corresponding envy-graph is acyclic.

- Give an item to an agent that no-one envies. In the previous step we showed that we can always find an allocation such that its corresponding envy-graph is acyclic. Therefore, there should be a vertex in the envy-graph with no incoming edges. Let i be the agent corresponding to this vertex. Since i has no incoming edges in the envy-graph, no other agent envies i. At this step, we ask agent i to pick one item among all remaining goods.

The following Lemma shows the approximation guarantee of our algorithm.

Lemma 3.3.6 *Suppose that we are given a partial α-EFX allocation $\mathcal{A}$ such that for every agent i and every remaining item b, we have $V_i(b) \leq \alpha' \cdot (\mathcal{A}_i)$ for some constant $\alpha' \leq 1$. Then, the resulting allocation after performing the method mentioned above is* $\min\{\alpha, \frac{1}{1+\alpha'}\}$-EFX.

Proof. The algorithm repeats the following steps until it allocates all items.

- Find and eliminate all the directed cycles from the envy-graph.
- Give an item to an agent that no-one envies.

Consider the step in which the algorithm eliminates cycles. As we discussed earlier, this step does not change the approximation factor of the algorithm. Hence, if the allocation is α-EFX before this step, it remains α-EFX after it (See *Lemma 6.1* in [58] for more detail).

Consider the second step of the algorithm. In this step, the algorithm finds an agent such that no-one envies this agent, and it allocates an item to this agent. Suppose our algorithm allocates item b to agent i. Since no-one envies agent i before this step, for every other agent j, we have $V_j(\mathcal{A}_i) \leq V_j(\mathcal{A}_j)$ where $\mathcal{A}$ is the allocation of items before this step. In addition we have $V_j(b) \leq \alpha' \cdot V_j(\mathcal{A}_j)$ since item b was among unallocated items at the beginning of this step. Thus, we have

$$V_j(\mathcal{A}_i) + V_j(b) \leq (1+\alpha') \cdot V_j(\mathcal{A}_j)\,.$$

This means that after allocation b, no agent j thinks the value of the bundle of agent i is more than $(1+\alpha') \cdot V_j(\mathcal{A}_j)$. Therefore, the allocation remains $\frac{1}{1+\alpha'}$-EFX. Hence, the final allocation is $\min\{\alpha, \frac{1}{1+\alpha'}\}$-EFX. □

Note that since EFX is a stronger notion of fairness than EFR, we can restate the lemma above for EFR allocations.

Corollary 3.3.7 *Suppose that we are given a partial α-EFR allocation $\mathcal{A}$ such that for every agent i and every remaining item b, we have $V_i(b) \leq \alpha' \cdot (\mathcal{A}_i)$ for some constant $\alpha' \leq 1$. Then, the resulting allocation after performing the method mentioned above is $\min\{\alpha, \frac{1}{1+\alpha'}\}$-EFR.*

We now show that at the beginning of Step 3, the valuation of every remaining item is small for all agents.

Observation 3.3.8 *efrsth Let $\mathcal{A}$ be the allocation after Step 2. Then for an agent i and every remaining item b we have*

- *If $i \in \mathsf{G}_1$, $V_i(b) \leq V_i(\mathcal{A}_i)/\varphi$.*
- *If $i \in \mathsf{G}_2$, $V_i(b) \leq V_i(\mathcal{A}_i)/3$.*
- *If $i \in \mathsf{G}_3$, $V_i(b) \leq V_i(\mathcal{A}_i)/3$.*

Proof. Consider an agent $i \in \mathsf{G}_1$, then by Observation 3.2.10 we have

$$V_i(b) \leq V_i(\mathcal{A}_i)/r_i \leq V_i(\mathcal{A}_i)/\varphi\,.$$

Next consider an agent $i \in \mathsf{G}_2$. This agent has two allocated items. Let $\mathcal{A}_i = \{b_i, b'_i\}$ be these items where b_i is the item allocated using NSW matching. Since agent i picks his best remaining item at Step 2 of our algorithm, we have

$$V_i(b) \leq V_i(b'_i)\,. \tag{3.7}$$

Also by Observation 3.2.10 we have

$$V_i(b) \leq V_i(b_i)/r_i \leq V_i(b_i)/2\,, \tag{3.8}$$

since $r_i > 2$ for agents in G_2. It follows from (3.7) and (3.8) that

$$V_i(b) \leq \big(V_i(b_i) + V_i(b'_i)\big)/3 = V_i(\mathcal{A}_i)/3\,.$$

The last case is when $i \in \mathsf{G}_3$. In this case agent i has three allocated items which are all larger than every remaining item. Therefore $V_i(b) \leq V(\mathcal{A}_i)/3$. $\square$

It follows from the observation above that for every agent i the valuation of every remaining item is at most $V_i(\mathcal{A}_i)/\varphi$ after the second step of our algorithm. Recall that our allocation by the end of Step 2 is $(3/4)$-EFR. Therefore, using Corollary 3.3.7, the allocation at the end of Step 3 is $\min\left\{\frac{3}{4}, \frac{1}{1+1/\varphi}\right\}$-EFR. Since $\varphi = 8/3$, it follows that our final allocation is $8/11 \approx 0.72$-EFR. This, coupled with the fact that all the steps can be implemented in polynomial time follows Theorem 3.1.1.

3.4 From NSW Matching to approximate EFX Allocation

In this section, we show that our idea to use NSW matching as the first step of the allocation can easily give a $(\phi - 1)$-EFX allocation where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. The approximation ratio of our algorithm matches the state-of-the-art $(\phi - 1)$ approximation result by Amanitidis et al. [67]. Likewise the algorithm in the previous section, our $(\phi - 1)$-EFX algorithm consists of 3 steps, namely NSW matching, allocation refinement, and envy-graph based allocation. The first and the third steps of our algorithm are almost the same as our previous algorithm. For the sake of completeness, we will briefly restate these steps in the rest of the section. The outline of our algorithm is represented in Algorithm 8.

Algorithm 8: The outline of the $(\phi - 1)$-EFX algorithm.

```
   // Step 1
1  $(\mathcal{A}, \mathcal{M}')$ = NSWMatchingAllocation($V$, $\mathcal{M}$);
2  $(\mathsf{G}_1, \mathsf{G}_2)$ = EnvyRankPartition($\mathcal{A}$);
   // Step 2
3  $\mathcal{O}$ = TopologicalOrderedAgents($\mathcal{A}$);
4  **foreach** $i \in \mathsf{G}_2$ *ordered by* $\mathcal{O}$ **do**
5      ($\mathcal{A}$, $\mathcal{M}'$) = ExtendAllocation($\mathcal{A}$, $\mathcal{M}'$, $i$);
6  **end**
   // Step 3
7  **while** $\mathcal{M}' \neq \varnothing$ **do**
8      Eliminate all directed cycles in the envy-graph;
9      Let $s$ be an arbitrary source in the envy graph;
10     ($\mathcal{A}$, $\mathcal{M}'$) = ExtendAllocation($\mathcal{A}$, $\mathcal{M}'$, $s$);
11 **end**
12 return $\mathcal{A}$;
```

3.4.1 Step 1.

In the first step, we allocate one item to each agent using NSW matching. Let $\mathcal{A}$ be the resulting allocation, and let $\phi = \frac{1+\sqrt{5}}{2}$ be the golden ratio. Based on the envy-rank of the agents, we divide them into 2 groups G_1 and G_2, as follows:

- Agent i belongs to G_1 if $r_i > \phi$.
- Agent i belongs to G_2 if $r_i \leq \phi$.

3.4.2 Step 2.

In the second step, we allocate one item to each agent in G_2 via the following process. let $\mathcal{O}$ be a topological ordering of the agents with respect to the envy-graph. We order the agents in G_2 according to $\mathcal{O}$ and ask them one by one to pick their most valuable remaining good.

After this step, the bundle of every agent in G_2 contains two items. For an agent $i \in \mathsf{G}_2$ we use $\{b_i, b'_i\}$ to denote the items allocated to this agent where b'_i is the item allocated in Step 2. We also use $\{b_i\}$ to denote the only item received by an agent $i \in \mathsf{G}_1$. We show that at the end of Step 2 the following clamis hold:

Claim 3.4.1 *efxstf By the end of Step 2, the allocation is* EFX *with respect to the agents in* G_1.

Proof. Let i be an agent in G_1, we show that for every other agent j we have

$$V_i(\mathcal{A}_i) \geq \max_{b \in \mathcal{A}_j} V_i(\mathcal{A}_j \setminus \{b\}) \,,$$

so the allocation is EFX from the agent i's perspective. If $j \in \mathsf{G}_1$, then we have $|\mathcal{A}_j| = 1$. Therefore,

$$\max_{b \in \mathcal{A}_j} V_i(\mathcal{A}_j \setminus \{b\}) = 0 \,,$$

and the claim clearly holds.

Consider an agent $j \in \mathsf{G}_2$. At the end of Step 2, agent j has two allocated items. By Observation 3.2.10, the valuation of the item b'_j for agent i is bounded by $V_i(\mathcal{A}_i)/r_i =$

$V_i(b_i)/r_i$. Therefore,

$$\begin{aligned}
\max_{b \in \mathcal{A}_j} V_i(\mathcal{A}_j \setminus \{b\}) &= \max\left\{V_i(b_j), V_i(b'_j)\right\} \\
&\leq \max\left\{V_i(b_j), V_i(b_i)/r_i\right\} && \text{Observation 3.2.10.} \\
&\leq \max\left\{V_i(b_i), V_i(b_i)/r_i\right\} && \text{Observation 3.2.9.} \\
&= V_i(b_i) = V_i(\mathcal{A}_i)\,. && r_i > 1.
\end{aligned}$$

Therefore the allocation is EFX. □

Claim 3.4.2 *efxsts By the end of Step 2, the allocation is* $(\phi - 1)$-EFX *with respect to the agents in* G_2.

Proof. Let i be an agent in G_2, we show that for every other agent j we have

$$V_i(\mathcal{A}_i) \geq (\phi - 1) \cdot \max_{b \in \mathcal{A}_j} V_i(\mathcal{A}_j \setminus \{b\})\,,$$

so the allocation is $(\phi - 1)$-EFX from agent i's perspective. If $j \in \mathsf{G}_1$, then we have $|\mathcal{A}_j| = 1$, and the claim clearly holds.

Consider an agent j in G_2. We first consider the case that $V_i(b_i) < V_i(b_j)$. In this case, the position of agent i in $\mathcal{O}$ is before agent j. Therefore, agent i receives his second good before agent j, and we have

$$V_i(b'_i) \geq V_i(b'_j)\,. \tag{3.9}$$

It follows that

$$\begin{aligned}
\max_{b \in \mathcal{A}_j} V_i(\mathcal{A}_j \setminus \{b\}) &= \max\left\{V_i(b_j), V_i(b'_j)\right\} \\
&\le \max\left\{V_i(b_j), V_i(b'_i)\right\} && \text{By (3.9).} \\
&\le \max\left\{r_j \cdot V_i(b_i), V_i(b'_i)\right\} && \text{Observation 3.2.9.} \\
&\le \max\left\{\phi \cdot V_i(b_i), V_i(b'_i)\right\} && r_j \le \phi. \\
&\le \phi \cdot \left(V_i(b_i) + V_i(b'_i)\right) \\
&= \phi \cdot V_i(\mathcal{A}_i)\,.
\end{aligned}$$

Therefore,

$$V_i(\mathcal{A}_i) \ge \frac{1}{\phi} \cdot \max_{b \in \mathcal{A}_j} V_i(\mathcal{A}_j \setminus \{b\})\,.$$

Since $\frac{1}{\phi} = \phi - 1$, it follows that our allocation is $(\phi - 1)$-EFX. The other case is when $V_i(b_i) \ge V_i(b_j)$. In this case, we have

$$\begin{aligned}
\max_{b \in \mathcal{A}_j} V_i(\mathcal{A}_j \setminus \{b\}) &= \max\left\{V_i(b_j), V_i(b'_j)\right\} \\
&\le \max\left\{V_i(b_i), V_i(b'_j)\right\} && V_i(b_i) \ge V_i(b_j). \\
&\le \max\left\{V_i(b_i), V_i(b_i)\right\} && \text{Observation 3.2.10.} \\
&= V_i(b_i) \le V_i(\mathcal{A}_i)\,.
\end{aligned}$$

Therefore, in this case the allocation is EFX which completes the proof of the claim. □

3.4.3 Step 3.

In the third step, we use the envy-graph to allocate the rest of the items. We show that at the beginning of Step 3, the valuation of every remaining item is small for all the agents.

Observation 3.4.3 *efxsth Let $\mathcal{A}$ be the allocation after Step 2. Then, for an agent i and every remaining item b we have*

- *If $i \in \mathsf{G}_1$, $V_i(b) \leq V_i(\mathcal{A}_i)/\phi$.*
- *If $i \in \mathsf{G}_2$, $V_i(b) \leq V_i(\mathcal{A}_i)/2$.*

Proof. Consider an agent $i \in \mathsf{G}_1$, then by Observation 3.2.10 we have

$$V_i(b) \leq V_i(\mathcal{A}_i)/r_i \leq V_i(\mathcal{A}_i)/\phi\,.$$

Next, consider an agent $i \in \mathsf{G}_2$. This agent has two allocated items which are larger than every remaining item. Therefore, $V_i(b) \leq V(\mathcal{A}_i)/2$. □

Therefore, for every agent i, the valuation of every remaining item is at most $V_i(\mathcal{A}_i)/\phi$. In order to complete the allocation we repeat the following steps until all the goods are allocated.

- Find and eliminate all the directed cycles from the envy-graph.
- Allocate an item to an agent that no-one envies to him.

Recall that our allocation by the end of Step 2 is $(\phi-1)$-EFX. Therefore, as we discussed in Lemma 3.3.6, the approximation ratio of our approach is $\min\{\phi - 1, \frac{1}{1+1/\phi}\}$. Since $\frac{1}{1+1/\phi} = \phi - 1$, it follows that our final allocation is $(\phi - 1) \approx 0.61$-EFX.

Chapter 4: Fair Allocation of Divisible Chores

4.1 Introduction

The *chore division* problem is the problem of fairly dividing an object deemed undesirable among a number of agents. The object is possibly heterogeneous, and hence agents may have different valuations for different parts of the object. Chore division was first introduced by Gardner [6] in 1970s, and is the dual problem of the celebrated *cake cutting* problem. In cake cutting, we would like to fairly divide a good (such as a cake) for which everyone has a positive valuation. In some sense, chore division is a minimization problem while cake cutting is a maximization problem. Recently [8] provided a bounded *envy-free* protocol for 4-person cake cutting, and later on a bounded envy-free protocol for n-person cake cutting [9]. Chore division or cake-cutting with negative utilities is less explored and much less is known about it. We provide the first *discrete and bounded envy-free chore division protocol for any number of agents*.

The fair cake-cutting problem was introduced in the 1940s. There are different ways one can define *fairness*. Initially, *proportional division* was studied. An allocation is *proportional* if everyone receives at least a $\frac{1}{n}$ fraction of the cake according to his/her valuation. Proportional division was solved soon in 1950 [16]. A stronger criterion of envy-freeness was proposed by George Gamow and Marvin Stern in 1950s, which is, *no* *one envies another*. In other words, each agent receives a part he thinks is the largest part.[1]. The envy-free cake cutting problem became "one of the most important open problems in 20th-century mathematics" according to Garfunkel [7].

For the case of two agents, the *"I cut you choose"* protocol simply provides an envy-free allocation for both cake cutting and chore division. However, the problem is highly more complicated for more agents. In general, since the valuations of agents for different parts of the object may be complex, the standard is to assume a query access model for evaluations. We can ask an agent its value for a part of the object, and also ask an agent to trim the object up to a certain value. For the case of three agents, Selfridge and Conway independently found an envy-free protocol for cake cutting. Oskui (see [69]) provided a solution for 3-person chore division, which is similar to Selfridge-Conway procedure for cake cutting, but is more complicated and needs 9 cuts instead of 5. Finding a finite protocol for cake cutting with more than three agents remained an open problem for a long time until [18] presented a *finite* envy-free protocol for cake cutting for any number of agents in 1995. Although this was a breakthrough in the field, their protocol is finite but *unbounded*, i.e., it does not guarantee *any bound* on the number of queries and even the number of cuts. Later Peterson and Su [74] provided an unbounded envy-free protocol for chore division. [75] and [76] gave "moving-knife" protocols for cake-cutting for four and five agents. [77] gave a moving-knife procedure for 4-person chore division. A moving-knife procedure involves one or more agents moving knives simultaneously with some restrictions until one agent calls "stop". Although moving-knife procedures are more than existence theorems, "a moving knife protocol is certainly less than an effective

[1]It is easy to see an envy-free allocation is also proportional

procedure in the algorithmic sense" according to [78]. That is because the continuous movement of a knife cannot be captured by any finite protocol.

Having a bounded envy-free protocol even for four agents remained an important open problem [79, 18, 68, 80, 81, 82, 83, 84, 85, 86, 69, 76]. The unboundedness of cake cutting protocols was mentioned as a "serious flaw" [86], and finding a bounded protocols was highlighted as "the central open problem in the field of cake-cutting" [84] and "one of the most important open problems in the field" [76]. Brams and Taylor [18] were aware of their protocol's drawback and explicitly mentioned "even for $n = 4$, the development of finite bounded envy-free cake cutting protocols still appears to be out of reach and a big challenge for the future".

Finally, the prominent work of Aziz and Mackenzie [8] provided a bounded envy-free cake cutting protocol for four agents. Later they generalized their work and provided an envy-free cake cutting protocol for *any* number of agents to settle a major and long-standing open problem. However, it remained an open problem to find a bounded envy-free chore division protocol even for $n = 4$. We provide the first discrete and unbounded envy-free protocol for chore division among any number of agents.

4.2 Prelimiaries

In chore division, we are asked to partition a given chore R among n agents. Let $A = \{a_1, \ldots, a_n\}$ be the set of agents, and $C_a(P)$ denote the cost of some piece $P \subseteq C$ for agent a. w.l.o.g we assume that For every agent a, the cost of the whole chore is 1, i.e., $c_a(C) = 1$. An envy-free partition is a partition of R into n pieces $P_1, \ldots, P_n$ and assigning them to the agents accordingly such that for every two agents a and b, $C_a(P_a) \leq C_a(P_b)$, where P_a and P_b denote the pieces assigned to agents a and b respectively.

For any protocol, we use the standard Robertson-Webb model [69]. In Robertson-Webb model, the chore is modeled as an interval $R = [0, 1]$. We have absolutely no knowledge about the agents' cost functions in advance, except that the functions are defined on sub-intervals of $[0, 1]$, non-negative, additive, divisible, and normalized. Therefore every information is obtained via queries. The complexity of a protocol is defined by the number of queries it makes. There are two types of information queries:

- $Trim_a(\alpha)$: given a cost value $0 \leq \alpha \leq 1$, agent a returns an $0 \leq x \leq 1$, such that his cost for interval $[0, x]$ equals α.
- $Eval_a(x)$: returns the cost value of interval $[0, x]$ for agent a.

In this proposal, we distinguish between cutting and trimming of a piece. Cutting a piece P refers to dividing P into two pieces, but in trimming we only find a subinterval in P and do not cut the piece. Note that in this proposal a piece P is not necessarily an interval, but a union of intervals, since we may cut and join pieces. Although $Eval$ and $Trim$ queries are defined on intervals, whenever we cut a piece we maintain the cost of the new pieces. Thus we can translate a query on a piece to a query on the interval $[0, 1]$.

4.3 Results and Techniques

Our main result is a discrete and bounded envy-free protocol for dividing a chore among n agents. Many techniques have been proposed for envy-free cake cutting. [9] provide a bright and powerful framework to obtain a bounded and envy-free protocol for cake cutting among n agents. However the components of their framework and their protocols do not work for chore division. The protocols for positive valuations are not usually applicable for negative valuations, and "in general there are no reductions from allocation to chores to goods or vice versa"[87]. To solve the chore variant of the problem, we borrow the general idea of their framework, but we have to provide novel techniques and structural results and also rebuild their framework's components. These new techniques and structures not only deliver powerful tools for designing chore division protocols, but also are useful in cake cutting.

In the following, we present the very high-level concepts and techniques used in our protocol. The basic idea is to use an inductive algorithm. More precisely we use induction on the number of agents and try to divide a chore only among a subset of the agents.

Initially, we need an envy-free protocol which partially divides a chore among the agents. The protocol does not necessarily allocate the whole chore, but roughly speaking assigns a fraction of the chore, maintaining the envy-freeness. The protocol has other plausible features to be mentioned later.

Having a partial allocation, we use the concept of *irrevocable advantage (dominance)*. It is the key of many fair allocation protocols [8, 9, 18, 76, 77, 74]. Assume that the partial allocation is envy-free and we have a remaining or unallocated chore R. We say an agent a has an irrevocable advantage to another agent b or a dominates b, if a thinks she is assigned much less chore than b, such that she may not envy b even if we assign the whole R to her. In other words, $C_a(P_b) - C_a(P_a) \geq C_a(R)$, where P_a and P_b are the pieces allocated to a and b respectively in the partial allocation. We use a similar but weaker notion of *significant advantage*. Agent a has a significant advantage over b if P_a is much more desirable than P_b to a with respect to the remaining chore, or more precisely $C_a(P_b) - C_a(P_a) \geq \alpha \times C_a(R)$, where α is a constant to be defined later. Importantly we show that significant advantage and irrevocable advantage are in some sense equivalent. If agent a has an irrevocable advantage over b, then her advantage is significant as well. On the other hand if agent a has a significant advantage over agent b, using some partial allocation protocols we make R small enough for agent a to make the advantage irrevocable.

Assume that we have a partial envy-free allocation. If there exists a set of agents $S \subset A$, such that each agent in S has irrevocable advantage to every agent in $A \setminus S$, we can leave $A \setminus S$ unchanged, and assign the remaining chore inductively to S. Thus the main goal of our protocol is to make a set of agents have significant/irrevocable advantage over the rest of the agents.

The other very useful concept, introduced by [9], is the notion of *snapshots*. Recall that we have a partial allocation protocol. Every time we may partially allocate the remaining chore to the agents. Each of these partial allocations is called a *snapshot*. The chore assigned to each agent is the union of her assigned chores in all the snapshots. A critical thing about snapshots is that we can use an agent's advantage in one snapshot to compensate her for modifications in other snapshots. Basically, if agent a has a lot of advantage over b in one snapshot, we can for example assign some of b's chore to a in some other snapshot. Also note that, as long as every snapshot is envy-free, if an agent a has irrevocable advantage to agent b, then she also has irrevocable advantage in total. Thus we can focus on one snapshot and deliver irrevocable advantage among some agents in that single snapshot. Then we can use other snapshots for having irrevocable advantage among other agents. Another very handy use of snapshots is that we may have as many of them as we need. Then we can concentrate on a set of similar snapshots. More precisely, in a snapshot every agent can order the other agents based on how much is their value for her allocated piece. [8] define two snapshots *isomorphic* if, roughly speaking, those orderings of the agents are exactly the same. Here we need a stronger notion of isomorphism. First, we define a *mask* of a snapshot, which somehow codes the significance of agents' advantages. We say two snapshots are isomorphic if each agent orders the other agents exactly the same and also their masks are the same. Having isomorphic snapshots, we can modify the allocated pieces easier, and thus we construct as many snapshots to be able to have a large enough set of isomorphic snapshots, using pigeon hole principle. We initially call this set of snapshots the *working set*. We set aside the other snapshots and only modify the working set.

The other useful concept that we introduce is a *matching*. A set of trimmed pieces and agents have a matching if we can match every trimmed piece to an agent such that the allocation is envy-free. We use this extra information about pieces to obtain more structural protocols. We show that if we have a matching we can define *monotone* protocols, which means we may only make the trimmed pieces larger, obtaining an envy-free allocation. We also use matching in the *Sub Core* protocol, in which we put a lower bound on the trims of pieces and try to trim the pieces and guarantee to maintain a matching.

Now we describe a technical overview of the main protocols. As aforementioned we need a partial envy-free allocation protocol called the *core* protocol. The Core protocol is the main and most fundamental protocol of our algorithm. The Core protocol has to have the following properties.

- Assigns each agent a piece such that no agent envies another agent;
- Assigns at least a $\frac{1}{n}$ fraction of the chore in one agent's point of view;
- Most importantly, given a specific agent, guarantees that this agent has significant advantage to another agent in this allocation.

The Core protocol is the "engine" of [8, 9]'s protocol for cake cutting, but unfortunately their protocols are not applicable for the chore division. Instead we design a much simpler Core protocol. Although our Core protocol is very simple, its proof is based on a much more complicated infinite protocol which guarantees the existence of the desired allocation. The basic idea of a Core protocol is as follows. We select a cutter agent that divides the chore into n equal pieces. Note that the pieces are not necessarily equal to other agents. Then we try to match each agent to one piece such that every agent receives a part of its matched piece, but at least one agent may be given a whole piece. Thus, at least $\frac{1}{n}$ fraction of the cake is allocated in the cutter's point of view. Also, the cutter receives some considerable advantage to the agent who has been given a whole piece. The heart of our Core protocol is the following structural lemma.

Lemma 4.3.1 *Given n pieces and n different agents, there exists an allocation of pieces to agents such that a whole piece is allocated to one agent, and a trim of each piece is allocated to exactly one agent, if and only if, there exists an ordering of the agents and an ordering of the pieces such that the following protocol provides an envy-free allocation. Agents receive their pieces one by one. The first agent receives the first piece. The i-th* *agent trims the i-th piece in such a way that she receives the largest part of it without envying the first $i-1$ agents. In other words she considers the first $i-1$ allocated pieces, if her cost for any of those pieces is less than her cost for the i-th piece, she trims the i-th piece to make it equal to that piece.*

Note that in such a protocol, the i-th agent may not envy the first $i-1$ agents, but some of the first $i-1$ agents may envy the i-th agent. Roughly speaking, this lemma shows that if there exists some "core-like" allocation of some pieces to agents, there exist an ordering of both agents and pieces such that the first $i-1$ agents also do not envy the i-th agent, using the aforementioned protocol. Thus, if there exists such allocation, we can try every ordering of agents and pieces to find an envy-free allocation using that simple protocol. Interestingly, we design a protocol which is even *infinite* but outputs a core-like allocation. However, knowing that there exists such a protocol is sufficient to be able to design a much simpler Core protocol.

The other important component of our protocol is the *Permutation* protocol. Assume that there is a set $S \subset A$ of agents, such that every agent in S has significant advantage to some agent $a \in A$ in a set of snapshots. If we could exchange the piece allocated to a with the allocated piece of some other agent $b \notin S$ in some snapshot, then every agent in S also has significant advantage to b. In Permutation we try to find such set S, that has significant advantage to a, and then somehow move the a's piece among every agent not in S. Therefore every agent in S has significant advantage to every agent in $A \setminus S$, and we can do the chore division inductively as we discussed. For exchanging the agents' pieces we find a chain of agents, $a_1, a_2, \ldots, a_k$, such that a_i receives a_{i-1}'s piece and a_1 receives

a_k's piece. Since each snapshot is envy-free after changing the pieces, agents $a_1, \ldots, a_k$ may envy each other or other agents. Thus we modify many other snapshots to guarantee envy-freeness. [9] also have a Permutation protocol. The key difference between our Permutation protocol is that in cake cutting we can add a piece of cake from R to a piece that is assigned to an agent, such that another agent accepts to receive it. However in chore division we have to remove a part of chore from a piece to be able to assign it to some other agent. The difference is huge and makes the Permutation much more subtle because of the two following reasons. First the part that we remove from a piece assigned to an agent goes back to the remaining chore, or R. Since R becomes larger, the significance of the advantages, which are defined based on R may change. Second, since the pieces that we want to remove are already allocated, it is not easy to divide them between agents, or remove similar pieces from other agents.

Moreover, we make use of two previously known fair division protocols. The protocols are used as infinite protocols for cake cutting, but we show that one can use them as powerful tools for bounded protocols as well. The first protocol is the *Near-exact* protocol introduced by [88]. In the Near-exact protocol, given a chore R, n agents, an integer m, and a real number $\epsilon > 0$, we divide E into m pieces, such that for each agent a and piece P, $|C_a(P) - \frac{1}{m}C_a(R)| \leq \epsilon C_a(R)$. In other words the pieces have almost equal costs for the agents. We also show how one can use the Near-exact protocol to improve [9]. The other protocol is the *Oblige* protocol, first used by [74]. In Oblige protocol we partition the chore into 2^{n+1} pieces, and output a partial envy-free allocation such that every agent is assigned at least one of the pieces completely. The combination of these protocols is used in our Discrepancy protocol, described below.

Another important component of our protocol is the Discrepancy protocol. We use the Discrepancy protocol when we have a piece P that is very costly for a set of agents S and R is relatively small, and in the contrary the rest of agents think R costs much more than P. Thus we use a combination of Near-exact and Oblige protocols to divide R among S and P among the rest of the agents, such that no agent envies another agent. In this way we may inductively divide the chore among smaller set of agents.

4.4 The Main Protocol

The Main Protocol is responsible for allocating the whole chore between agents in an envy-free manner. It tries to make a set of agents dominant to others and then allocates the remaining chore between a smaller number of agents. The main protocol achieves this goal by using two other protocols, the Core Protocol, and the Permutation Protocol. Given an agent a as the cutter, the Core Protocol asks a to divide the chore into n equally preferred pieces and then partially allocate the chore between agents such that at least one piece is completely allocated and each agent gets part of a single piece. The Permutation Protocol gets a partial allocation of the chore and tries to slightly modifies it to make a set of agents dominant to others.

At first, the Main Protocol calls the Core Protocol many times, each time on the remaining chore to create a large number of partial allocations. We call each of these partial allocations a *snapshot*.

Definition 4.4.1 *A* ***snapshot*** *s is a partial envy-free allocation returned by Core Protocol. We use s^a to denote the allocated piece to the agent a.*

After generating many snapshots, Main Protocol finds a set of similar snapshots and edit these snapshots in Permutation Protocol. Each time we call the Core protocol, we get an envy-free partial allocation of the chore. In each of these snapshots, each agent thinks that the cost of her piece is less than the cost of others'. In particular, considering a snapshot s and agents a and b, since the partial allocation obtained by Core Protocol is envy-free, agent a thinks that cost of piece s^a is not greater than cost of the s^b. We define the *advantage of agent a over agent b* in this snapshot, the amount of chore that a thinks that she got less than agent b, i.e:

$$Adv^s_{a,b} = C_a(s^b) - C_a(s^a)$$

If the advantage that agent a has over b is greater than the cost of the residual chore, agent a will not envy b, no matter how the residual chore will be allocated between agents. In this case, we say that agent b is dominated by agent a. In Particular agent a dominates agent b in the partial allocation s if $Adv^s_{a,b} \geq C_a(R)$.

Since in Core Protocol the cutter cuts the chore into n equal pieces according to her own perspective and the protocol allocates at least one piece completely, the cost of the residual chore is at most $\dfrac{n-1}{n}$ for the cutter. In Permutation Protocol we modify a set of similar snapshots such that if we reduce the size of chore by calling the Core Protocol nB_n times with each agent as the cutter B_n times where $B_n = n^n$, then we can find set of agents B such that each agent in A dominates every other agent in $A \setminus B$. Therefore, if the advantage of agent a over another agent b is at least $C_a(R) \times (\frac{n-1}{n})^{B_n}$ during the Permutation Protocol, this agent will dominate agent b after reducing the size of chore.

We define a value to be *significant* for agent a if it is at least $C_a(R) \times (\frac{n-1}{n})^{B_n}$ and otherwise *insignificant*. Here a key idea is that if agent a has a significant advantage over agent b, we can reduce the size of the remaining chore so that a dominates b. In Permutation Protocol, we mainly try to modify snapshots such that it gives a significant advantage to a set of agents over all other agents.

Since a significant value could become insignificant or vice versa by slightly modifying the residual chore or allocated pieces, we need large the gap between significant and insignificant values to make sure that a significant value remains significant if we only slightly modify the allocated pieces and R. To this end, we define very significant and very insignificant values as follows:

Definition 4.4.2 *A value v is* ***very significant*** *for an agent if it is at least 2^{2n} times the cost of R in her perspective.*

A value v is ***very insignificant*** *for an agent if R costs at least 2^{2n} times more than v.*

Aziz and Mackenzie in [9] show that we can large the gap between significant and insignificant values using a bounded number of quires by calling the Core Protocol many times.

At the beginning of Main Protocol, we run the Core Protocol $IS_n \times n^{n^{n^n}}$ times where $IS_n = n^{n^n}$. Our goal is to find a set of IS_n similar snapshots. In each run, we set the first agent as the cutter and partially allocate the residual chore between agents. Lets s_i be the snapshot generated in the i^{th} call of Core Protocol. The following claim shows that in each snapshot the cutter has a significant advantage over some other agent.

Claim 4.4.3 *In each snapshot returned by the Core Protocol, the cutter has a significant advantage over at least one other agent.*

Proof. In the beginning of Core Protocol, the cutter a divides the residual chore R into n equally preferred pieces. The Core Protocol returns a partial envy-free allocation such that at least one piece is completely assigned to a single agent and update the residual chore. If the cutter got this complete piece, then she thinks that $1/n$ of the chore is allocated to her. Since the allocation is envy-free every other piece which is allocated to other agents costs at least $1/n$ for the cutter. Hence the remaining chore has a cost equal to zero to her, and based on the definition she has a significant advantage over all other agents. Otherwise, assume that the complete piece is allocated to another agent. Since the partial allocation is envy-free, the advantage of the cutter over this agent is at least $\dfrac{C_a(R)}{n-1}$. Since $C_a(R)(\dfrac{n-1}{n})^{B_n} \leq \dfrac{C_a(R)}{n-1}$ the cutter has a significant advantage over the agent who got the complete piece. □

After generating snapshots, in each snapshot s, for every agent a, we ask a to place a trim on any piece other than s_a to make it equal to her piece if cost of this piece is not significantly larger than s_a in her perspective. Main Protocol passes these trim lines to Permutation Protocol which uses the trim lines to modify the allocated pieces and make them desirable for other agents, and then tries to exchange the pieces between agents. An important observation is that if in a snapshot s an agent a has a significant advantage over some other agent b, and we give s_b to some other agent c while preserving the envy-freeness, then a got a significant advantage over c in s.

We use $t_1^{s,a}, t_2^{s,a}, \cdots, t_{l_{s,a}}^{s,a}$ to denote the trim lines from right to left on the piece s_a where s is an arbitrary snapshot and a is an arbitrary agent, and $l_{s,a}$ is number of agents who had trim on this piece. In the same way we will denote agents who have trims on this piece from right to left by $d_1^{s,a}, d_2^{s,a}, \cdots, d_{l_{s,a}}^{s,a}$. Moreover, we can partition each piece based on trim lines. We use $e_1^{s,a}, e_2^{s,a}, \cdots, e_{l_{s,a}}^{s,a}$ to partition s_a, where $e_i^{s,a}$ is a part of s_a between two consecutive trims which the left trim is $t_i^{s,a}$.

Permutation Protocol detaches some part of the pieces from trim lines. We want the cost of all detached pieces be very small for all agents, so that very significant advantages remain significant after this procedure. To this end, we make sure that every $e_i^{s,a}$ costs either very significant or very insignificant for all the agents. For this purpose, while there is an agent who thinks at least one part is neither very significant or very insignificant, we keep reducing the size of the residual chore for this agent by calling Core Protocol. After that for every part $e_i^{s,a}$, we define *mask* of this piece or $mask_i^{s,a}$ to be the set of agents who think this part costs very significantly. After that, if we find a part $e_i^{s,a}$ which costs very significantly to agent b, and trim line of this agent lies on the left of $t_i^{s^a}$, we can say that agent b has got a very significant advantage over agent a in this snapshot, and remove trim on this agent from s^a.

The set of snapshots given to Permutation Protocol should have very similar properties. In particular, the protocol needs that the order of trims on each piece be the same between different snapshots and every part has the same mask in all the snapshots.

Definition 4.4.4 *We call two snapshots s and s'* ***isomorphic*** *if :*

- *For every pieces s_a and s'_a, they have the same number of trims on them and order of agents who have a trim on these pieces be the same in both snapshots.*
- *For every part $e_i^{s,a}$ and $e_i^{s',a}$, the mask of these parts be the same.*

In the following lemma, we show that if we generate at least $IS_n \times n^{n^{n^n}}$ snapshots, then we can find at least IS_n isomorphic snapshots. The Main Protocol finds these isomorphic snapshots and passes them to the Permutation Protocol.

Lemma 4.4.5 *Every set of $IS_n \times n^{n^{n^n}}$ has at least IS_n isomorphic snapshots.*

Proof. Considering a piece in a snapshot, this piece has at most $n-1$ trims on it, and for each part of the piece between two trims, there are 2^n possible ways to choose the set of agents who think this part costs very significant. So the number of possibilities one piece can have is:

$$\sum_{k=0}^{n-1} \binom{n-1}{k} k! \times 2^{n^k} \leq n! \times 2^{n^n}$$

this number is less than n^{n^n} for $n \geq 3$.

Since there are n pieces in a snapshot, the number possibilities a snapshot could have is $n^{n^{n^n}}$. So, among $IS_n \times n^{n^{n^n}}$ snapshots, there are at least IS_n isomorphic snapshots. $\square$

4.5 The Core Protocol

Aziz and Mackenzie in [9] present a Core Protocol as the core engine of their discrete and bounded algorithm for the cake cutting problem. In each call of the Core Protocol, they allocate some cake from the residue to all the agents in an envy-free manner. By each call of this protocol, they make the remaining cake smaller, but there is no guarantee that calling this protocol for bounded times suffices to allocate all the cake in an envy-free manner. Nonetheless, they use this protocol several times in different parts of their main algorithm.

In the chore division problem, we have a Core Protocol, Algorithm 10, for allocating additional chore from the residue to all the agents in an envy-free manner. Our Core Protocol works as follows: First we ask the specified cutter to cut the chore into n equal pieces $p_1, p_2, \ldots, p_n$ according to her own perspective. Then, for each ordering of the agents and each ordering of the pieces, we make a new allocation of the pieces to the agents. In the new allocation, agents receive their pieces one by one. The first agent receives the first piece. The i^{th} agent trims the i^{th} piece in such a way to equalize it with her most preferred piece among the first $i-1$ allocated pieces (we consider the cost value of each piece from its leftmost side to its trim.) If this allocation be envy-free we return the allocation. In Lemma 4.5.15, we guarantee an ordering of agents and ordering of pieces exists such that the protocol returns an envy-free allocation. In Subsections 4.5.1 through 4.5.7, we provide another core protocol, which is not bounded but we use it to guarantee such an ordering of the agents and the pieces exists. For this reason, we call the new core protocol *Existential Core Protocol*, Algorithm 11.

4.5.1 Existential Core Protocol

We call our Existential Core Protocol on set of agents A with one specified cutter, and unallocated chore R. In the first step of the protocol we ask the cutter to cut the chore into n equal pieces $p_1, p_2, \dots, p_n$ according to her own perspective. From now, we work on these n pieces, and we frequently ask the agents to make trims on them. In different steps of the algorithm, we may have many trims on each piece, but we have one specific trim that we call it the *main trim*. We may change the position of the main trim on a piece, but we always have exactly one main trim on each piece. As we mentioned before, our Existential Core Protocol does not necessarily allocate whole of the chore to the agents, and it finally allocates each of these pieces from their leftmost side to their main trim. Initially the main trim of each piece is on its rightmost side, and we change their place frequently during the algorithm. In each step of the algorithm we may allocate a piece up to its main trim to only one agent. It is very crucial to note that, in this section, when we say we allocate a piece to an agent we mean that it is allocated from its leftmost side to its main trim. Also, when we ask the cost value of a piece from a specific agent, she reports her cost value from the leftmost side of the piece to its main trim.

In Algorithm 11, after the cutter cuts the chore, we run the Separated Chore Core

Protocol on all the agents and all the pieces with their main trims. The Separated Chore Core Protocol receives n pieces of the chore with their main trims and a set of n agents, and it returns an envy-free partial allocation of the pieces to the agents. The properties of Separated Chore Core Protocol are as follows:

Definition 4.5.1 *(Separated Chore Core Protocol Properties)*

- *It does not change the main trim of pieces to a righter position.*
- *It does not change the main trim of at least one of the pieces.*

In Algorithm 11, when we call Separated Chore Core Protocol, all the main trims are on the rightmost side of the pieces, but we make many other calls on Separated Chore Core Protocol such that the main trims are not necessarily on the rightmost side of the pieces. Separated Chore Core Protocol guarantees that its returned allocation does not change the main trim of at least one piece. Therefore, we can imply that from the cutter's perspective, at least $1/n$ of the chore is allocated. Existential Core Protocol, Algorithm 11, returns the allocation that Separated Chore Core Protocol returned. In the following Lemma we prove that if Separated Chore Core Protocol works, Existential Core Protocol works as well.

Lemma 4.5.2 *If Separated Chore Core Protocol, Algorithm 12, works, Existential Core Protocol, Algorithm 11, gives an envy-free partial allocation to n agents in which one of the agents is the cutter who cuts the chore into n pieces, each agent gets a part of one of the pieces, and at least one agent gets a complete piece.*

Proof. If Algorithm 12 works correctly, its returned allocation is an envy-free partial allocation, and it does not change the main trim of at least one of the pieces. Since in Algorithm 11, we call Separated Chore Core Protocol for the pieces with a main trim on the rightmost side, at least one of the pieces is completely allocated in the returned allocation by Separated Chore Core Protocol. □

4.5.2 Separated Chore Core Protocol

In this Subsection we describe Separated Chore Core Protocol, Algorithm 12. As we mentioned in Subsection 4.5.1, this protocol receives a chore with n pieces as well as a set of n agents, and it returns an envy-free partial allocation of the pieces to the agents such that the main trim of at least one of the pieces remains intact. We say a piece is *intact* during a protocol P if its main trim does not change during the call of P.

This protocol is based on an iterative idea in lines 2-21 of Algorithm 12. After the i^{th} iteration of the loop we ensure that we have a neat allocation for the first i agents. We define a *neat* property for allocations as follows:

Definition 4.5.3 *We call an allocation of m disjoint pieces of the chore to n agents (where $n \leq m$)* ***neat*** *if the following properties hold:*

- *The allocation allocates a (not necessarily whole) part of exactly one of the pieces to each agent.*
- *no agent prefers an unallocated piece or another agent's allocation to her allocation.*

In the i^{th} step of the loop, before running Line 20, we already have a neat allocation of pieces to the first i agents (we describe it in details later), and in Line 20, by running Best Piece Equalizer Protocol, it modifies the neat allocation. Best Piece Equalizer Protocol, Algorithm 15, is a protocol receiving a neat allocation of some pieces to some agents (one piece each agent), and it returns a modified neat allocation of pieces to the agents (one piece each agent). The properties of Best Piece Equalizer Protocol are as follows:

Definition 4.5.4 ***(Best Piece Equalizer Protocol Properties)***

- *The protocol is monotone.*
- *The returned allocation does not have any subset of bad agents.*

We define the monotonicity of a protocol as follows:

Definition 4.5.5 *Assume that P is a protocol which receives a neat allocation of pieces to agent set A as input, and outputs another neat allocation of the pieces to the same set of agents. We call protocol P* ***monotone*** *if and only if it does not change the main trim of any piece p to a lefter position.*

We also define a bad subset of agents as follows:

Definition 4.5.6 *When we have a neat allocation of the pieces to some of the agents, we call a subset of agents S* ***bad*** *if the following conditions hold:*

- *One piece is allocated to each agent in S.*
- *None of the allocated pieces to the agents in S is intact.*

- *For each agent $a \in S$, the cost of the piece that we have allocated to a is less than the cost of any other piece.*

We describe Best Piece Equalizer Protocol more briefly in Subsection 4.5.5.

Now, we describe how the protocol makes a neat allocation of pieces to the first i agents before running Best Piece Equalizer Protocol in Line 20. In the i^{th} step of the loop, agent a_i chooses piece p which is her most preferred piece among all pieces. However, p may be allocated before. If in the end of the $(i-1)^{th}$ step of the loop, p is not allocated to any agent, then we can simply allocate it to a_i (As mentioned before, we emphasize that when we allocate a piece to an agent, we allocate a partial part of it from its leftmost side to its main trim). Although we easily handled the case that p is not allocated, the other case is much harder. If p has been allocated to another agent a_j before, we have a conflict of interest on piece p. We define a *popular* piece and its *happy* or *sad fan* agents as follows:

Definition 4.5.7 *When at least two agents a and b prefer a specific piece p to all other pieces, we call p a **popular piece**, and we call agents a and b the **fan**s of piece p. We also call agent a a **happy fan** of p if she is a fan of p and p is already allocated to her, and we call her a **sad fan** of p if she is a fan of p but p is not already allocated to her.*

According to Definition 4.5.7, piece p is a popular piece, agent a_j is its happy fan, and agent a_i is its sad fan. We handle this conflict of interest based on two different cases whether the main trim of p is the same its initial main trim or not. First, we deal with the case that the main trim of p is not changed. In this case, we run Allocation Extender Protocol for all the pieces with their main trims, the first i agents with their current allocation, the popular piece p, and its fan agents a_i and a_j. Allocation Extender Protocol is a protocol which receives a neat allocation of pieces to the agents, a popular piece p, and its two specific happy and sad fan agents a and b. Note that in the allocation that this protocol receives, agent b is the only agent who does not have any piece. This protocol returns a neat allocation of pieces to the agents such that every agent has a piece and the main trim of piece p is not changed during the call of the protocol. We describe Allocation Extender Protocol more briefly in Subsection 4.5.3.

Now, we deal with the case that p is not an intact piece. The general idea to handle this case is that to modify the current allocation such that an intact piece becomes the popular piece. Then, we can handle it similar to the previous case. We do this modification by calling Core Match Refiner Protocol, Algorithm 17. Core Match Refiner Protocol is a protocol which receives a neat allocation of pieces to agents, with a specific popular piece p. This protocol returns a new neat allocation and a flag variable with the following properties:

- In the new allocation piece p does not have any owner agent.
- If flag be false, the new allocation has assigned a piece to each called agent.
- If flag be true, the new allocation has assigned a piece to each called agent except one of them, who is the sad fan of one of the intact pieces.

We call Core Match Refiner Protocol for all the pieces with their main trims, the first $i-1$ agents with their current allocation, and the popular piece p. The call returns a refined neat allocation and a flag (In the case the flag is true, it returns the new popular piece q with agent a as its happy fan and agent b as its sad fan.) In the new allocation, piece p does not have any owner agent, so we can easily allocate it to a_i. If the flag is false, we have already increased the size of our neat allocation. If the flag is true, the situation is similar to the previous case such that there exists a popular intact piece q. Similar to the first case, we can run Allocation Extender Protocol (Line 16 of the Algorithm 12).

In the following lemma, we prove the correctness of Separated Chore Core Protocol.

Lemma 4.5.8 *If Allocation Extender Protocol, Core Match Refiner Protocol, and Best Piece Equalizer Protocol work, Separated Chore Core Protocol, Algorithm 12, gives an envy-free partial allocation to the called agents such that Separated Chore Core Protcol Properties hold.*

Proof. This protocol works based on an induction on the number of agents. We use a stronger induction hypothesis to prove that the protocol works correctly. We prove that after the i-th step of the loop, we have a neat allocation of pieces to the first i agents such that at least one of the agents receives an intact piece, and there exists no bad subset of agents.

As the base case of the induction, in the first step of the loop, since all the pieces are unallocated, we can easily allocate the most preferred piece of a_1 to her without changing its main trim. Therefore, we have a neat allocation to the first agent who receives an intact piece, and there exists no bad subset of agents.

Now, we assume that after the first $i - 1$ step of the loop, we have a neat allocation of pieces to the first i agents such that at least one of the agents receives an intact piece, and there exists no bad subset of agents. In the i-th step, if we can allocate the most preferred piece of a_i, piece p, to her, since after the $i-1$-th step, we had a neat allocation, we still have a neat allocation, and one of the agents still have an intact piece. By running Best Piece Equalizer Protocol, since it is a monotone protocol, we do not change the main trim of the allocated intact piece. Moreover, running Best Piece Equalizer Protocol makes sure that we do not have any bad subset of agents in our neat allocation. Therefore, the induction works for this case.

Now, we deal with the case that piece p has been allocated to a_j, and we cannot allocate it to a_i. In this case, if p be an intact piece, we run Allocation Extender Protocol. This protocol returns us a neat allocation of pieces to all the first i agents, and it does not change the main trim of p. Therefore, we have a neat allocation such that at least one of the agents receives an intact piece. Now, assume that p is not an intact piece. In this case, first we run Core Match Refiner Protocol, and then we allocate p to a_i (Since after the $i-1$-th step of the loop, we do not have any bad subset of agents, we can run Core Match Refiner Protocol.) If flag be false, we already have a neat allocation, and Core Match Refiner Protocol has assigned an intact piece to one of the agents. If flag be true, we have converted this case to the case that the popular piece was intact. Therefore, by running Allocation Extender Protocol, we have a neat allocation of pieces to all the first i agents. In the end, running Best Piece Equalizer Protocol returns a refined neat allocation without any bad subset of agents. As we mentioned, since this protocol is monotone, its result does not change the main trim of an intact piece. Hence, we can infer that after the last step of the loop, we have a neat allocation of pieces to agents such that at least one of the pieces is intact, and there exists no bad subset of agents. □

4.5.3 Allocation Extender Protocol

In this subsection, we describe Allocation Extender Protocol, Algorithm 13. This protocol receives a neat allocation of the pieces to the agents such that all the called agents has a piece except agent b who is the sad fan of the popular piece p. The goal of this protocol is that to find a neat allocation which allocates a piece to each called agent and does not change the main trim of p.

In this protocol, first we ask agents a and b to trim each piece equal to p. For each piece q, we change its main trim to the rightmost trim made by a and b. Then, we deallocate all the allocated pieces but we remember the allocation as the *original mapping* and the main trim of the pieces as the *original mapping trims*. Then we run SubCore Protocol, Algorithm 14. SubCore Protocol is a protocol which receives a set of agents and pieces with an original mapping such that the main trim of each piece is not righter than its original mapping trim. It returns a neat allocation of pieces to agents (one piece each agent) such that the following properties hold:

Definition 4.5.9 ***(SubCore Protocol Properties)***

- *It does not change the main trim of unallocated pieces.*
- *It does not change the main trim of any allocated piece to a lefter position.*
- *It does not change the main trim of a piece to a righter position than its original mapping trim.*

We describe this protocol more briefly in Subsection 4.5.4. We emphasize that we do not change the original mapping during each call of SubCore Protocol. We call

SubCore Protocol for all the first i agents except a and b with all the pieces except piece p. This call gives us a neat allocation of the called pieces to the called agents. After this call, we allocate a piece to a and another piece to b from the unallocated pieces in the following manner. We should have at least two unallocated pieces such that one of them is p. We take one of the other unallocated pieces q. First, we allocate q to the agent among a and b who made the main trim on it, and then, we allocate p to the other agent. Now, we have allocated a piece to each agent. In the following lemma, we prove that Allocation Extender Protocol works correctly.

Lemma 4.5.10 *If SubCore Protocol works, Allocation Extender Protocol returns a neat allocation of pieces to agents (one piece each agent) such that* p *is an intact piece in this protocol.*

Proof. In the beginning of the Algorithm, agents a and b make trims on each piece equal to p. They can make this trim because p is their most preferred piece. Then, by calling SubCore Protocol, we have a neat allocation of pieces to all agents except a and b. Then, as we mentioned, we allocate an unallocated piece to each agent a and b. Without loss of generality, assume that agent a receives p and agent b receives q. Since SubCore Protocol returns a neat allocation to the called agents, They do not envy each other. They also do not envy to a, b or any unallocated piece, because after running SubCore Protocol, we have not changed the main trim of pieces. agents a and b do not envy the other agents, because according to SubCore Protocol Properties, SubCore Protocol does not change the main trim of allocated pieces to a lefter position, and it does not change the main trim of unallocated pieces. Agents a and b do not envy to an unallocated piece, because SubCore

Protocol does not change the main trim of unallocated pieces. It is also easy to check that they do not envy each other. □

4.5.4 SubCore Protocol

In this Subsection, we describe Algorithm 14 in more details. As we mentioned before, this protocol gets a subset of agents and a subset of pieces with their main trims as well as the original mapping of agents to pieces. As we mentioned, in the original mapping, we have an allocation of pieces to agents (one piece each agent, and some of the pieces may be unallocated) such that in each call of the protocol, for each allocated piece in the original mapping, its original mapping trim is righter than its main trim. The protocol returns a neat allocation of pieces to the agents such that SubCore Protocol Properties, definition 4.5.9, hold.

In this protocol, similar to the idea that we used in Algorithm 12, we find a neat allocation iteratively. In Algorithm 14, we implement this loop in lines 2-32. We add the agents one by one, and in the end of the i-th step of the loop, we have a neat allocation of pieces to the first i agents. Similar to Algorithm 12 in the i-th step, we ask agent a_i to choose her most preferred piece. Assume that p is her choice, and it is not allocated to other agents. In this case, we can easily allocate p to a_i. Otherwise, we should handle the case in a much more complicated manner.

When agent a_i chooses a piece which is allocated to another agent, we have i agents that the first preference of each of them is among $i-1$ pieces. We call this set of allocated pieces P. We have $|P| = i-1$ allocated pieces, and we call these $i-1$ pieces *contested pieces*. The intuition to resolve this issue is to find a way to allocate one of the pieces outside of P to one of the first i agents. However, none of the first i agents may prefer these pieces. Hence, for using this idea, we need to change the main trim of some contested pieces and reallocate them to agents. To this end, we ask each of the first i agents to make a trim on each of the contested pieces to equalize it with the cost value of her most preferred piece outside of P. For an agent, if the cost value of a contested piece from its leftmost side to its original mapping trim was less than the cost value of her most preferred piece outside of P, She makes a trim on the original mapping trim of the contested piece. We define a representative agent for each piece in P (we have not assigned them yet.) If the trim of agent $a_j \in \{a_1, a_2, \ldots, a_i\}$ on contested piece q was the rightmost trim, and q was the original mapping of a_j, we set a_j as the representative of q. Otherwise, the agent with the rightmost trim and lowest index is its representative. We define set W as the set of agents who are the representative of at least one piece in P. In Lemma 4.5.11, we prove that W is the set of agents who are guaranteed to have a neat allocation. Therefore, our idea is to enlarge $|W|$ up to $|P|$ and make sure we have a neat allocation of P to agents in W. In each step of the loop in lines 19-30 of Algorithm 14, we increase the size of W by adding exactly one agent to it. In each step of this loop, we update the main trim of each piece in P by agents in $W' = \{a_1, \ldots, a_i\} \setminus W$ as follows: for each piece $p \in P$, we find the rightmost trim among the trims that the agents in W' have made on p, and update the main trim of p to this trim. Then, we find another mapping of pieces to agents in W' (one piece each agent), and we call it *modified original mapping*. We will use this mapping as the original mapping in our recursive call of SubCore Protocol. We find the modified original mapping as follows: we know that each agent in W is a representative of at least one piece in P. For each agent $a_j \in W$, if $p_{a_j} \in P$ and a_j was the representative of p_{a_j}, we assign p_{a_j} to a_j in the modified original mapping. Otherwise, we assign an arbitrary piece, from the set of pieces that a_j is their representative, to her. In both cases, the *modified original mapping trim* is the trim of a_j on the piece. The *modified original mapping trim* of a piece is its main trim in the modified original mapping. Moreover, the modified original mapping trim of each unallocated piece of P in the modified original mapping is the trim of its representative agent on it. In Lemma 4.5.11, we prove that this is an envy-free mapping of pieces to agents.

After finding the modified original mapping of W to P, we call SubCore Protocol on all pieces in P with their main trims, the agents in W, and the modified original mapping of W recursively. This call allocates a piece of P to each agent in W in a neat manner. If $|W| = |P|$, we break the loop, and among the first i agents, we have exactly one agent that we have not allocated any piece to her. Let a be this specific agent. We allocate the most preferred piece out of P for a to her. In Lemma 4.5.11, we prove that this allocation is feasible. Now, as the other case, assume that $|W| \neq |P|$. In this situation, we should further continue enlarging W. We take an arbitrary piece q from unallocated pieces of P. Let agent $a \in \{a_1, \ldots, a_i\} \setminus W$ be the agent who has the rightmost trim on q (In Lemma 4.5.11, we prove that this trim is the main trim of q.) If more than one such agent exists, a is the agent with the lowest index. We allocate q to a and add a to W. Note that in the beginning of this loop we ignore the previous trims of agents in W and deallocate the allocated pieces, because we will find another allocation by calling SubCore Protocol recursively.

In the following lemma, we prove that our SubCore Protocol guarantees a neat allocation, and all the SubCore Protocol Properties hold.

Lemma 4.5.11 *Suppose that we have set A of n agents and m pieces where $n \leq m$. Each of the pieces has an original mapping trim as well as a main trim such that the main trim of each piece is not righter than its original mapping trim. Also, assume that there is a neat allocation for the agents and the pieces with their original mapping (using their original mapping trims). Then, calling the SubCore Protocol for these agents and pieces returns a neat allocation of pieces to agents such that all the SubCore Protocol Properties that we mentioned in Definition 4.5.9 hold.*

Proof. The SubCore Protocol works based on an induction on the number of agents n. As the base case, when $n = 1$, the protocol allocates the most preferred piece of the only agent to her, and it returns the allocation without changing any main trim. The allocation is clearly neat, since the agent chooses her most preferred piece, and all the SubCore Protocol Properties hold, since we do not change any main trim. Therefore, the Lemma for $n = 1$ works.

For n agents, Algorithm 14 makes a neat allocation of the pieces to agents iteratively by the loop in Lines 2-32. We call this loop the *main loop* of the protocol. The main idea of the proof is as follows: in the end of the i-th iteration of the main loop, we make sure that we have a neat allocation for the first i agents such that all the SubCore Protocol Properties hold. For the first agent in the main loop, her most preferred piece can be easily allocated to her, since none of the pieces has been allocated yet. Hence, after the first iteration, we have not changed any main trim, and for the first agent, all the SubCore Protocol Properties hold. Now, assume that after the $i-1$-th iteration, for the first $i-1$

agents, we have a neat allocation such that all the SubCore Protocol Properties hold. In the i-th step of the main loop, the algorithm first asks a_i to choose her most preferred piece p. We have two cases, and we discuss about them separately: (1) p is not allocated to an agent and (2) p is allocated to an agent.

Case (1) is the easier case. In this case, we allocate p to a_i tentatively. Since after the $i-1$-th iteration of the main loop, the allocation for the first $i-1$ agents was neat, after allocating p to a_i, they do not envy a_i. Moreover, since a_i chooses her most preferred piece, she does not prefer any other agent's or any unallocated piece. Hence, our allocation is a neat allocation. In this step of the loop, we do not change any main trim, thus all the SubCore Protocol Properties hold.

Now, we deal with case (2). In this case, in our final allocation, we use the same set of allocated pieces, P, as well as one of the pieces out of P. However, we may change the allocation, and some agents may not receive the same piece that they had before. First we reallocate P to $i-1$ agents of the first i agents in the loop in Lines 19-30. Then, we allocate one of the pieces out of P to the only remaining agent among the first i agents. First, we prove that the properties of the SubCore Protocol hold. Line 23 is the only place we may update the main trims during our protocol. In this update, we only may change the main trim of the pieces in P, and we change them to a righter position. As we discussed before, we allocate all the pieces in P by the end of the i-th iteration of the main loop. Hence, we do not change the main trim of unallocated pieces, and we do not change the main trim of the allocated pieces to a lefter position. Therefore, we can infer that all the SubCore Protocol Properties hold.

Now, we prove that we find a neat allocation by the end of case (2). We initialize set W with the representatives of the pieces in P, and we increase its size one by one in the loop in Lines 19-30. First of all, we prove that running SubCore Protocol recursively is possible in this loop. To this end, we prove that the modified original mapping has two properties:

- **Property (A1)**: For each agent $a \in W$, if in the mapping, q be the assigned piece to agent a, the modified original mapping trim of q is righter than its main trim.
- **Property (A2)**: The modified original mapping is envy-free.

If we prove the above properties, calling SubCore Protocol in Line 24 is feasible.

Proof of Property (A1): Since $q \in P$, q is the most preferred piece among all the pieces for at least one agent a in the first i agents. Therefore, the trim of a is not lefter than the main trim of q, and it implies that the rightmost trim on q is not lefter than its main trim.

Proof of Property (A2): For each agent $a \in W$, we show that in the modified original mapping, she does not envy any other agent $b \in W$. Let p_1 and p_2 be the pieces that we have assigned to agents a and b in the modified original mapping consecutively. As the first case, let the trim of agent a on p_2 be lefter than the trim of agent b on it. In this case, clearly a does not envy b. As the second case, let the trim of both agents a and b be lefter than the original mapping trim of p_2. In this case, a does not envy b as well. The reason is that in this case the cost value of p_2, from its leftmost side to the trim of a, for a is equal to the cost value of her most preferred piece outside of P, and the cost value of p_1, from its leftmost side to the trim of a, is at most equal to the cost value of her most preferred piece outside of P. As the last case, assume that the trim of both agents a and

b on p_2 are on the original mapping trim of p_2. We deal with this case in two following subcases:

- **SubCase 1**: p_1 is the assigned piece to agent a in the original mapping.
- **SubCase 2**: p_1 is not the assigned piece to agent a in the original mapping.

Analysis of SubCase 1: Since the original mapping is envy-free, in this SubCase agent a does not envy agent b.

Analysis of SubCase 2: Let $q \neq p_1$ be the assigned piece to agent a in the original mapping. If $q \in P$, since the trim of agent a on piece p_2 is on its original mapping trim, her trim on piece q should be on its original mapping trim as well. Therefore, agent a should be the representative of q, and the protocol should assign q to a in the modified original mapping instead of p_1. Therefore, $q \notin P$. Now, if q is the most preferred piece for agent a outside of P, since the original mapping allocation is envy-free, the cost value of q should be equal to the cost value of both p_1 and p_2 (for each of these pieces, by the cost value of a piece, we mean its cost value for agent a from its leftmost side to the trim of a.) In this situation, we can imply that agent a does not envy to b. Now, assume that q is not the most preferred piece for agent a outside of P. Since the original mapping is envy-free, the cost value of q, from its leftmost side to its original mapping trim, for a is not more than the cost value of p_2, from its leftmost side to its original mapping trim. Since the main trim of q is not righter than its original mapping trim, and q is not the most preferred piece for a outside of P, the trim of a on p_2 should be lefter than its original mapping trim. This is not feasible, because in this case, we assumed that the trim of both agents a and b are on the original mapping trim of p_2.

The proof of Property (A2) is complete.

Immediately after calling SubCore Protocol, the agents in set W do not envy each other, because we have recursively called the SubCore Protocol for smaller number of agents, and according to the induction hypothesis, we can say that SubCore Protocol returns a neat allocation. It also infers that the agents in W do not prefer the unallocated pieces in P to their piece. Now, we prove that any agent $a \in W$ does not prefer any piece out of P to her own piece. To this end, assume that the allocated piece to agent a after calling SubCore Protocol is piece p_1. According to SubCore Protocol Properties, the main trim of p_1 is not righter than its modified original mapping trim. Therefore, if the modified original mapping trim of piece p_1 is made by agent a, she does not prefer any piece outside of P to p_1. Now, we deal with the case the modified original mapping trim of piece p_1 is made by another agent. We know that agent a has made the modified original mapping trim of at least one of the pieces in P. Let p_2 be this piece. According to SubCore Protocol Properties, we know that after calling SubCore Protocol the main trim of p_2 is not righter than its modified original mapping trim. Therefore, we can infer that since agent a does not prefer p_2 to p_1, she does not prefer the pieces outside of P as well.

After calling SubCore Protocol recursively, we have two following cases.

- **Case A**: $|W| \neq |P|$
- **Case B**: $|W| = |P|$

Analysis of Case A: If Case A happens, we take an arbitrary piece q from unallocated pieces of P such that $a \in W'$ is the agent with the lowest index among the agents who have the rightmost trim on q. Then, we allocate it to a and add a to set W. First of all, we show that the main trim of q is equal to the trim of agent a on it. We know that immediately after $i-1$-th iteration of the main loop, for each piece in P, at least one agent among the first $i-1$ agents prefer the piece the most. Assume that immediately after $i-1$-th iteration of the main loop, agent b is the agent who prefers q the most. Therefore, when the agents trim the pieces in P, agent b does not trim q on a lefter position than its main trim. Now, if $b \notin W$, we can make sure that we update the main trim of q by a in Line 22. As the other case, assume that $b \in W$. In this situation, by calling SubCore Protocol, in Line 24, we have allocated a piece q' to b. According to SubCore Protocol Properties, by calling SubCore Protocol, we have not changed the main trim of q' to a lefter position. Therefore, since b does not prefer q to q', we can make sure that we update the main trim of q by a in Line 22.

Now, we show that agent a does not envy to the other agents in W. For the sake of contradiction assume that a envies another agent $b \in W$. Assume that after calling SubCore Protocol, in Line 24, piece p_1 is allocated to b. Before calling SubCore Protocol agent a has made her trim on piece p_1. Since before calling SubCore Protocol $a \notin W$, according to SubCore Protocol Properties, when we call SubCore Protocol, we do not change the main trim of p_1 to a lefter position than the trim of agent a. Therefore, the only way that a envies b is that the trim of agent a be on the original mapping trim of p_1. In this situation, the cost value of p_1, from its leftmost side to its original mapping trim, for agent a is less than the cost value of her most preferred piece outside of P, from its leftmost side to its main trim. Now, assume that p_2 is the allocated piece to agent a in the original mapping. p_2 should be in P or outside of P. Therefore, if we show that it can be in none of them, it is a contradiction. First, we show that $p_2 \notin P$. If $p_2 \in P$, since the original mapping is envy-free, the trim of agent a on p_2 is on its original mapping trim. Therefore, agent a should be the representative of p_2. It means that before calling SubCore Protocol, agent a should be in W, which is a contradiction. Now, we show that p_2 is not outside of P. We know that the cost value of p_1, from its leftmost side to its original mapping trim, for a is less than the cost value of her most preferred piece outside of P, from its leftmost side to its main trim, and also we know that the original mapping trim is envy-free. Therefore, we can infer that the cost value of p_2, from its leftmost side to its original mapping trim, for a is less than the cost value of her most preferred piece outside of P, from its leftmost side to its main trim. Since the main trim of p_2 is not righter than its original mapping trim, we can say that the cost value of p_2, from its leftmost side to its main trim, for a is less than the cost value of her most preferred piece outside of P, from its leftmost side to its main trim. This is a contradiction, because p_2 is in the outside of P. According to this contradiction, we can infer that agent a does not envy any agent $b \in W$. Moreover, since agent a updated the main trim of piece q, which is the piece that we allocated to her, she does not prefer the pieces outside of P to q as well.

Analysis of Case B: If Case B happens, we break the loop. Let a be the only agent in the first i agents that we have not allocated any piece to her. We allocate her most preferred piece outside of P to her. Let q be this piece. Clearly, agent a does not prefer any other piece outside of P to q, because q is the most preferred one. With a similar argument that we made in the analysis of Case A, we can show that agent a does not envy the agents in W.

In the above analyses we guaranteed the envy-freeness of our allocation. $\square$

4.5.5 Best Piece Equalizer Protocol

In this Subsection, we describe Best Piece Equalizer Protocol. Assume that we have a neat allocation of pieces to some agents. The goal of Best Piece Equalizer Protocol is that to update the neat allocation and the main trims of the allocated pieces such that all the Best Piece Equalizer Protocol Properties, Definition 4.5.4, hold. This protocol is based on a loop in Lines 1-17. We call it as the *main loop* of the protocol. In each step of the main loop we find a bad subset of agents, and by changing the current allocation, we make it a non-bad subset. This loop ends when we do not have anymore bad subset. Assume that S is a bad subset of agents in a step of the main loop, and P is the set of pieces that we have allocated to S. While S is a bad set, we ask each agent $a \in S$ to makes a trim on each piece $p \in P$ to equalize it with her most preferred piece out of P (If the cost value of p from its leftmost side to its rightmost side was less than the cost value of her most preferred piece out of P, she makes a trim on the rightmost side of p.) We call the leftmost trim on p which is righter than its main trim as the *equalizer trim* of p, and we call the current allocation of pieces to agents as the *old allocation*. We run Separated Chore Core Protocol on all the agents in S and all the pieces in P with their equalizer trims as their main trims. The call returns an envy-free partial allocation of pieces to agents such that all Separated Chore Core Protocol Properties that we mentioned in Definition 4.5.1 hold. We call the returned allocation the *new allocation*. In the new allocation, the main trim of a piece may change to a lefter position than its initial main trim. This makes our protocol non-monotone. We guarantee the monotonicity of the protocol by running Monotonicity Saver Protocol. Monotonicity Saver Protocol is a protocol receiving n agents as well as n

pieces, a neat allocation of pieces to agents (one piece each agent) as the *old allocation*, and a neat allocation of pieces to agents (one piece each agent) as the *new allocation*. This protocol returns another neat allocation such that the main trim of each piece in the result allocation is not lefter than the main trim of the secondary allocation. We run Monotonicity Saver Protocol for the agents in S, the pieces in P, and the old as well as the new allocation. We update our current allocation with the returned allocation by Monotonicity Saver Protocol.

In the following lemma, we prove the correctness of Best Piece Equalizer Protocol if Separated Chore Core Protocol and Monotonicity Saver Protocol work correctly.

Lemma 4.5.12 *If Separated Chore Core Protocol and Monotonicity Saver Protocol work correctly, Best Piece Equalizer Protocol returns a neat allocation (one piece each agent), and all the Best Piece Equalizer Protocol Properties, that we mentioned in Definition 4.5.4, hold.*

Proof. First of all, we prove that in each iteration of the main loop we make subset S of the agents a non-bad subset. When we ask each agent to make a trim on each piece p in P to equalize it with her most preferred piece out of P, at least the trim of the owner of p is righter than its main trim, because S is a bad subset of agents, and according to the definition, the cost value of each piece for its owner in P should be less than her most preferred piece out of P. Thus, we can guarantee that the two following properties:

- **Property 1**: The equalizer trim of each piece exists.
- **Property 2**: Each agent in S has a trim on at least one of the pieces which is not lefter than the main trim of that piece.

Then, by running Separated Chore Core Protocol, according to its properties in Definition 4.5.1, we receive an allocation such that at least one of the called pieces is intact, and the main trim of any other piece is not changed to a righter position. Before running Separated Chore Core Protocol, we have an allocation, and we call it the old allocation. After running Separated Chore Core Protocol we have another allocation, and we call it the new allocation. Now, we prove that the allocation that we have after running Montonicity Saver Protocol is neat, but before that let S_1 be the subset of S such that the main trim of their pieces have not changed by the Monotonicity Saver Protocol, and $S_2 = S \setminus S_1$.

- proof of envy-freeness in S: Since the returned allocation by Monotonicity Saver Protocol is envy-free partial, the agents in S do not envy each other.
- proof of envy-freeness of agents in S to agents out of S and unallocated pieces: As we mentioned, Separated Chore Core Protocol returns an allocation such that none of the main trims has changed to a righter position. Thus, according to Property 2, we can infer that in the allocation returned by Separated Chore Core Protocol for each agent a the cost value of at least one of the pieces in P is not more than the cost value of her most preferred piece out of P. Based on this and envy-freeness of the returned allocation by Separated Chore Core Protocol, we can infer that after running Separated Chore Core Protocol and before running Monotonicity Saver Protocol, the agents in S do not prefer any piece out of P to their allocated pieces. Since after running Monotonicity Saver Protocol the allocated pieces to S_1 (and their main trims) have not changed, the agents in S_1 do not envy to agents out of

S and unallocated pieces. Moreover, in the Monotonicity Saver Protocol, we have changed the allocated pieces to S_2 to their allocated pieces in the old allocation. In the old allocation, they did not prefer the pieces out of P to their own piece. Therefore, they do not prefer these pieces again.

- proof of envy-freeness of agents out of S to agents in S: the main trim of allocated pieces to agents in S_1 have not changed to a lefter position, and the main trim of pieces allocated to agents in S_2 have not changed. Therefore, since before running Separated Chore Core Protocol the agents out of S did not envy the agents in S, after running Monotonicity Saver Protocol, we keep this envy-freeness property.

By running Monotonicity Saver Protocol, we make sure that after each iteration of the main loop, the main trim of any piece has not changed to a lefter position. Therefore, we can infer that our protocol is monotone.

Unfortunately, the main loop of our protocol may iterate infinite times, but the good property of our protocol is that in each iteration, it changes the main trim of some pieces (at least one piece) to a righter position, and it does not change the main trim of any piece to a lefter position. The reason that in the main loop, we change the main trim of at least one of the pieces to a righter position is that all the equalizer trims are righter than the main trims, and by running Separated Chore Core Protocol at least one of the called pieces remains intact. Since each piece has a rightmost side, a limit, and our protocol is monotone, and it changes the main trim of at least one piece in each iteration, we can infer that we converge to a neat allocation solution if the protocol does not finish in finite iterations. $\square$

4.5.6 Monotonicity Saver Protocol

In this subsection, we describe Monotonicity Saver Protocol, Algorithm 16. As we mentioned in Subsection 4.5.5, this protocol receives a set of n pieces, a set of n agents, and two different allocation of pieces to agents with different main trims. We call these allocations the new and the allocations. Our goal is to change the new allocation such that the main trim of each piece does not be lefter than its main trim in the old allocation.

Let P be the set of pieces whose main trim in the new allocation is lefter than its main trim in the old allocation, and let S be the set of agents who own the pieces in P. First, we deallocate the pieces that we have allocated to the agents of S in the new allocation. Then, for each agent $a_i \in S$, we change the main trim of p_{a_i} to its main trim in the old allocation, and we allocate it to a_i. In Lemma 4.5.13, we prove that this allocation is feasible and the protocol works correctly.

Lemma 4.5.13 *Monotonicity Saver Protocol returns an envy-free partial allocation of pieces to agents (one piece each agent) such that the main trim of each piece in the returned allocation is not lefter than its main trim in the old allocation.*

Proof. If the reallocation that we made during the protocol be feasible, clearly the main trim of each piece in the returned allocation is not lefter than its main trim in the old allocation. The reason is that we changed the main trim of each piece whose main trim was lefter than its main trim in the old allocation. Now, first we prove that our reallocation was feasible, and then we prove that the returned allocation is envy-free partial.

Let S' be the set of agents that we have allocated P to them in the old allocation. If

$S = S'$, it means that the reallocation is feasible. Clearly, $|S| = |S'|$, and if for each agent $a \in S'$ we prove $a \in S$, it infers that $S = S'$. Since the old allocation is an envy-free partial allocation, for each agent $a \in S'$, the main trim of the piece that we allocated to her in the new allocation should be lefter than its main trim in the old allocation. Therefore, we can imply that $a \in S$.

Now, we prove that the returned allocation is envy-free partial. The agents of S do not envy each other, because the old allocation was envy-free. The agents of $A \setminus S$ do not envy each other, because the new allocation was envy-free. The agents of S do not prefer a piece whose main trim is righter than its main trim in the old allocation, thus they do not envy the agents of $A \setminus S$. Finally, the agents of $A \setminus S$ do not prefer a piece whose main trim is righter than its main trim in the new allocation, thus they do not envy the agents of S. □

4.5.7 Core Match Refiner Protocol

In Core Match Refiner Protocol, Algorithm 17, we build a special graph that we call it the *Core Match Refiner Graph*. We build Core Match Refiner Graph G as follows: For each piece we have a vertex in G. We connect the vertex of piece p to the vertex of piece q with a directed edge if and only if $C_a(p) = C_a(q)$ where agent a is the owner of p.

In the following lemma, we prove that Core Match Refiner Protocol works correctly.

Lemma 4.5.14 *Core Match Refiner Protocol works correctly.*

Proof. This lemma is almost trivial. The only point is that we can find a path P which ends in an unallocated or an intact piece. We can find P because we do not have any bad subset of agents and we have at least one unallocated or intact piece. $\square$

4.5.8 Final Remarks

In the previous subsections, for each protocol P we prove that if each of the protocols that P calls it works correctly, P works correctly as well. In Figure 4.1, we have a graph of these protocols. Protocol P_1 has an edge to Protocol P_2, if P_1 calls P_2. If the graph did not have a loop, we could infer that the whole procedure works correctly. However, we have two loops in the graph. The first loop happens in the SubCore Protocol such that it calls itself recursively. Anytime this protocol calls itself, the call happens on a smaller subset of agents, and we know that when the number of agents is one, it does not call itself anymore. The second loop happens between Separated Chore Core Protocol and Best Piece Equalizer Protocol. Similarly, each time Best Piece Equalizer Protocol calls Separated Chore Core Protocol, the call happens on a smaller number of agents. Moreover, Separated Chore Core Protocol does not call Best Piece Equalizer Protocol on a larger number of agents. We also know that when the number of agents is one, Separated Chore Core Protocol does not call Best Piece Equalizer Protocol anymore. Therefore, based on a simple induction we can infer that none of these loops makes any issue in the correctness of the whole procedure.

Running Existential Core Protocol returns an envy-free partial allocation. As we mentioned in Subsection 4.5.5, we may need infinite number of queries in each call of Best Piece Equalizer Protocol. This makes Existential Core Protocol unbounded, but

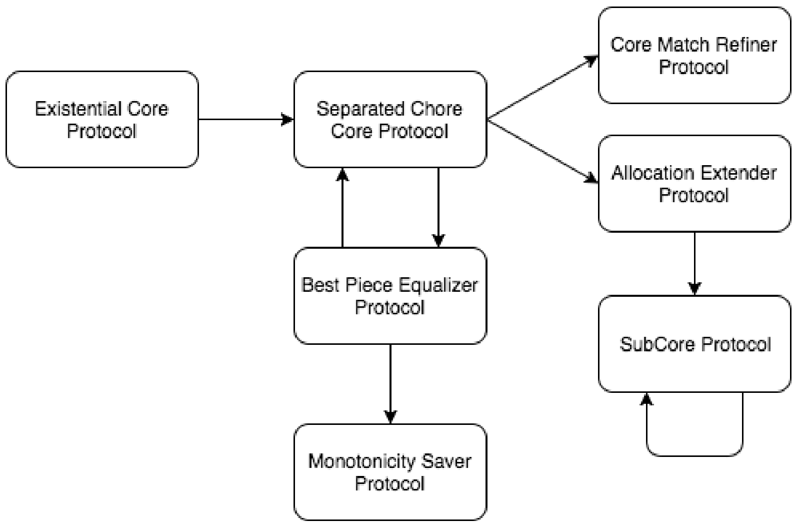

Existential Core Protocol
Separated Chore Core Protocol
Core Match Refiner Protocol
Allocation Extender Protocol
Best Piece Equalizer Protocol
SubCore Protocol
Monotonicity Saver Protocol

Figure 4.1: The Graph of the protocols in the Core. If Protocol P_1 has an edge to Protocol P_2, it means that P_1 calls P_2.

since Best Piece Equalizer Protocol is a monotone protocol with an upper-bound, as we discussed in Subsection 4.5.5, it converges to a solution.

Now, in the following lemma using Existential Core Protocol, we guarantee that Core Protocol, Algorithm 10, returns an envy-free partial allocation.

Lemma 4.5.15 *Core Protocol, Algorithm 10, is a discrete and bounded algorithm that returns an envy-free partial allocation of pieces to agents (one piece each agent) such that at least one of the pieces is completely allocated.*

Proof. The algorithm is clearly discrete and bounded. Now, we show that there exists an ordering of agents and pieces which guarantees an envy-free partial allocation. It suffices to show that the returned allocation of Existential Core Protocol can be made by one of the orderings of agents and pieces in Core Protocol. To this end, we analyze the properties of the returned allocation by Existential Core Protocol. Existential Core Protocol guarantees that we have allocated a complete piece to at least one of the agents. Without loss of generality, assume that agent a is this agent, and piece p is allocated to her. We create two vectors V_A and V_P of the agents and the pieces consecutively. They finally will be the ordering of the agents and the pieces that we are looking for. We add agent a to V_A and piece p to V_P. In the end of each iteration of the main loop of Separated Chore Core Protocol we call the best piece equalizer protocol. Therefore, in the returned allocation we should not have any bad subset of agents. First of all, for each agent $b \neq a$ who has received a complete piece q, we add b to V_A and q to V_P. Now, we iteratively add the other agents and pieces to V_A and V_P. Since there exists no bad subset of agents, at least one of the agents outside of V_A should prefer one of the pieces in V_P as the same as her own piece. We add this agent to the end of V_A and its piece to the end of V_P. We do this iteratively until we do not have anymore agent outside of V_A. Now, we claim that the ordering of V_A and V_P guarantees an envy-free allocation in Core Protocol, Algorithm 10. Since the returned allocation of Existential Core Protocol is envy-free, the allocation is envy-free. Now, for each piece, we should make sure the trim that an agent makes on it in Core Protocol, Algorithm 10, in the ordering of V_A and V_P is equal to its main trim in the returned allocation by Existential Core Protocol. The completely allocated pieces in the allocation of Existential Core Protocol are the first pieces in V_P. In Core Protocol, in the ordering of V_A and V_P, the agents trim the rightmost side of these pieces similar to Existential Core Protocol. For the other pieces, in Core Protocol, the agents trim the pieces to equalize them to their most preferred piece among the previously allocated pieces. In the ordering of V_A and V_P, in the returned allocation by Existential Core Protocol, we also know that for the incomplete pieces, their owners prefer at least one of the previously allocated pieces as much as their own piece. It implies that we have the same trims on the returned allocations of both protocols for the incomplete pieces as well.

Moreover, we should mention that the first agent in any ordering of agents in Algorithm 10 receives a complete piece. Thus, we have allocated a complete piece to at least one agent. □

4.6 Permutation Protocol

In the Permutation Protocol, we are given a large number of isomorphic snapshots. This protocol modifies these snapshots such that a set of agents $B \subset A$ get a significant advantage over other agents and it returns B. The Main Protocol use the result of permutation protocol to make every agent in B dominant to every agent in $A \setminus B$. After that, the residual chore can be allocated among the agents in B without worrying about agents in $A \setminus B$.

The Permutation Protocol iteratively finds set of agents $b_1, b_2, \cdots, b_k$ such that everyone won't envy others if agent b_i give his allocated piece in all the snapshots to the agent $b_{(i-1) mod k+1}$. Since envy-freeness won't preserve by giving the allocated piece of one agent to the other agent, we have to modify some snapshots to maintain envy-freeness. To this end, during our work, we will reserve some snapshots and whenever we want to swap the allocated pieces between agents, we modify the reserved snapshots to preserve envy-freeness. We declare *working set* to be the set of snapshots that protocol is working with them and use W to denote it. Note that the protocol will remove many snapshots from the working set and reserve them for the further modifications.

An important property of the Permutation Protocol is that it may modify some allocated pieces by detaching small part of them and the protocol ensures that for each modification, the cost of the detached piece is very insignificantly in every agent's perspective. Therefore, in case an agent a had a very significant advantage over some agent b in some snapshot, if we can modify the agents b's piece and giving it to some other agent c, then agent a would have a significant advantage over agent c, and later the Main Protocol use the significant advantage that a has over c, to make him dominant over c.

Recall that in the main protocol every agent is placed a trim on every other agent's piece to make it equal to his piece. Since the snapshots are isomorphic, the order of the trims is the same in all the snapshots. Moreover, we always modify all the snapshots in the working set in the same way to preserve isomorphism.

For an agent a we use p_a to denote agent a's allocated piece in all the snapshots and $t^a_1, t^a_2, \cdots, t^a_{l_a}$ to denote the trim lines from right to left on p_a where l_a is the number of trims on p_a and we use $d^a_1, d^a_2, \cdots, d^a_{l_a}$ to indicate the agents who had the trim on p_a from right to left.

Moreover, we can partition p_a based on the trim lines. We use e^a_i to denote the part of p_a which is between two consecutive trims and the left trim is t^a_i. Recall that, The Main Protocol ensures that for every e^a_i, the mask of this part is the same across all the snapshots, which means the agents who think the cost of this part is very significant are the same in all the snapshots.

These trim lines are somehow the guidelines for modifying the allocation. Consider an arbitrary piece p_a. If this piece has no trim on it, it means that all other agents have a significant advantage over this agent, so we are done. Otherwise, consider the rightmost trim of this piece which is t_1^a, if we cut p_a from this trim, then agent d_1^a is willing to exchange his piece for p_a. This is a basic idea behind the Permutation Protocol. It iteratively tries to detach some part of allocated pieces so that it finds a set of agents who are willing to exchange their pieces between themselves.

At each iteration, Permutation Protocol creates an empty graph G with n nodes. It adds a directed edge from node a to b if the next trim on the agent a's piece is for b. We add directed edges from a to all other nodes in case that all the trim lines on the agent a's piece are already detached. Recall that an agent places a trim on other agent's piece if the piece she has is not significantly smaller than the other agent's piece. Therefore if a piece has less than $n-1$ trims and we detach all of them, the set of agents who did not have a trim on this piece get a significant advantage over all other agents since every other agent has this piece in some reserved snapshot, and the protocol can return this set of agents. Therefore we can assume that all the pieces which all their trims are detached, had $n-1$ trim lines. We call these pieces *inactive* and others *active*.

Definition 4.6.1 *We call a piece* ***inactive*** *if it has $n-1$ trims and all its trim lines are already detached. We call every other piece an* ***active*** *piece. We call a node in the graph* *inactive if its corresponding piece is inactive and otherwise active.*

Considering an inactive piece, all the agents had this piece at some point. We can easily make this piece desirable for every other agent by reattaching some of the detached parts. Therefore, we can add directed edges from inactive pieces to all other nodes. Note that the Core Protocol gives a significant advantage to the cutter over some other agent, so at least one piece has less than $n-1$ trim lines. Similar to the approach used in [9] it is easy to show that the graph always contains a cycle with at least one active node.

Consider that we have found a cycle in graph G with at least one active node and we want to detach the next part of corresponding piece of every active node. By detaching some part of a piece and giving it to another agent, some other agents may envy. In general, if $e^a_1, \cdots, e^a_k$ are already detached from p_a and we detach e^a_{k+1} and give this piece to d^a_{k+1}, agents a and $d^a_1, \cdots d^a_k$ may envy d^a_{k+1} since the cost of this piece will be less than what they have. Therefore, whenever we want to detach some part of a piece, we also detach part of other agents' pieces to preserve envy-freeness.

Recall that the snapshots in the working set are not necessary envy-free, but we have modified some reserved snapshots such that no agent envies others. Therefore, a reserved snapshot or a snapshot in the working set may not be envy-free, but if an agent a envies another agent in a snapshot, this means that in other snapshots we have already detached some part of a piece that a has, and the cost of detached piece is larger than the amount that a envies others. For each snapshot, we define *envious* of a agent, to be the total amount that this agent envies others. We also say that a modification of a snapshot is good if it does not increase the envious of any agent. Therefore it preserves envy-freeness considering all the snapshots.

Definition 4.6.2 *Consider a not necessary envy-free allocation of m pieces $P = \{p_1, \cdots, p_m\}$ and set of m agents $A = \{a_1, \cdots, a_m\}$ such that piece p_i is allocated to agent a_i. For an agent a_i, we call the* ***envious*** *of this agent in this allocation, the total amount that agent a_i envies others which is:*

$$Env^P_{a_i} = \sum_{j=1}^{m} max\{0, C_{a_i}(p_i) - C_{a_i}(p_j)\}$$

Lets suppose that for each piece p_i there is a trim t_i on it. We say that a modification is a ***good modification*** *if :*

- *For each piece p_i it is cut upto its trim t_i.*
- *Exactly one piece is allocated to every agent in A after changing the allocation.*
- *If the cut pieces were $P' = \{p'_1, \cdots, p'_m\}$, then for each agent a_i, $Env^{P'}_{a_i} \leq Env^{p}_{a_i}$.*

It is obvious that the original allocation is a good modification for itself, i.e if we don't cut any piece and don't change the allocation, then it is a good modification. In the following lemma we show that there is good modification which is bounded and at least one piece is cut from its trim.

Lemma 4.6.3 *Consider the allocation of m pieces $P = \{p_1, \cdots, p_m\}$ and set of m agents $A = \{a_1, \cdots, a_m\}$ such that piece p_i is allocated to agent a_i. Lets suppose that for each* *piece p_i there is a trim t_i on it. There is a bounded algorithm which give us a good modification of pieces in P such that at least one piece is cut from its trim.*

Proof. For each pair of agents a_i and a_j, we create a virtual piece $v_{i,j}$ such that it costs $max\{0, C_{a_i}(p_i) - C_{a_i}(p_j)\}$ for agent a_i and 0 for all other agents. Lets create new set of set of pieces $P' = \{p'_1, p'_2, \cdots, p'_m\}$ by attaching the virtual pieces to left of the pieces in P, $p'_i = v_{1,i} + v_{2,i} + \cdots + v_{m,i} + p_i$. Every agent believes that the new allocation is envy-free. Formally, considering agents a_i and a_j, the amount that agent a_i envy agent a_j is:

$$\begin{aligned} C_{a_i}(p'_i) - C_{a_i}(p'_j) &= C_{a_i}(v_{i,i}) + C_{a_i}(p_i) - C_{a_i}(v_{i,j}) - C_{a_i}(p_j) \\ &= C_{a_i}(p_i) - C_{a_i}(p_j) - max\{0, C_{a_i}(p_i) - C_{a_i}(p_j)\} \leq 0 \end{aligned}$$

Cut each piece from its trim, then call the Subcore Protocol, giving it the pieces P' with set of agents A and original mapping $M = \{p'_1, p'_2, \cdots, p'_m\}$ to get new pieces $P'' = \{p''_1, p''_2, \cdots, p''_m\}$ where p''_j is a the piece p'_j after changing its main trim. Since the original mapping was envy-free, the protocol gives us an envy-free allocation of pieces such that at least one piece is intact. Therefore this piece is completely cut from its trim line.

Consider an agent a_i and suppose that p''_j is allocated to him. Since the allocation is envy-free, we have $C_{a_i}(p''_j) \leq C_{a_i}(p''_k)$ for all k. Now we remove the virtual pieces and show that what remains is a good modification of the original allocation. Since the virtual pieces were completely on the left side of trims, they are still attached to the pieces in P''.

Due to envy-freeness, we have:

$$\sum_{k=1}^{m} max\{0, C_{a_i}(p''_j) - C_{a_i}(p''_k)\} = 0$$

If we remove virtual pieces, the envious of agent a_i would become:

$$Env_{a_i}^{p''} \leq \sum_{k=1}^{m} max\{0, C_{a_i}(p''_j) - C_{a_i}(p''_k)\} + \sum_{k=1,k\neq j}^{m} v_{i,k} \leq \sum_{k=1}^{m} v_{i,k} = Env_{a_i}^{p}$$

□

Suppose that we want to detach a part of a piece p. We relabel the agents in order that they had a trim line on this piece, so that agent a_1 was the first agent who had this piece, agent a_2 detached a first part and so on. Lets suppose that trim line of agent a_k is already detached and agent a_k is the owner of this piece and we want to detach the next part of this piece which is $e_{k+1}^{a_1}$ and give it to the agent $d_{k+1}^{a_1}$ which is a_{k+1} since we have relabeled the agents. Let A' be set of agents who had this piece which is $A' = \{a_1, \cdots, a_k\}$. If we detach this part and give it to agent a_{k+1}, agents in A' may envy this agent. For an agent a_i in A', suppose that we have k set of reserved snapshots $S_1, S_2, \cdots, S_k$ such that size of every set is at least $|W|$, and in all the snapshots in S_j, the agent a_j has the piece p and in each snapshot in S_j, $e_{k+1}^{a_1}$ is more costly for a_i than cost of $e_{k+1}^{a_1}$ in any other snapshot in working set, i.e, for a snapshot s in S_j and s' in working set, we have :

$$C_{a_i}(e_{k+1}^{s,a_1}) \geq C_{a_i}(e_{k+1}^{s',a_1})$$

. We merge all the snapshots in $S_1, S_2, \cdots, S_k$ in the following way:

- For an agent a_j in A', we first concatenate the agent a_j's pieces in $S_1, S_2, \cdots, S_{i-1}, S_{i+1}, \cdots, S_k$. Considering the snapshots in S_j, they are isomorphic, therefore we can remove $e^{a_1}_{k+1}$ parts from a_j's pieces in these snapshots and concatenate the remaining to right of the a_j's merged piece and then add all the $e^{a_1}_{k+1}$ parts to right of the merged piece.

- For all other agents, we simply concatenate all this agent pieces in snapshots in $S_1, S_2, \cdots, S_k$.

We call this merging, the *piecewise merging*.

The main property of piecewise merging is that for an agent a_j in A' it keeps all the $e^{a_1}_{k+1}$ parts of a_j's pieces on the rightside of the merged piece. Therefore if we detach those parts from merged piece only agents in A' may envy a_j. To preserve envy-freeness between agents in A', we use lemma 4.6.3. We attach virtual pieces to agents $a_1, a_2, ..., a_k$ pieces and completely detach one of the $e^{a_1}_{k+1}$ from one agent in A' while preserving the envy-freeness.

Suppose that agent a_j is the agent that his piece is cut from its trim line. The cost of parts that are detached from a_j's piece is more than the sum of the cost of all $e^{a_1}_{k+1}$ parts in the working set. Since the modification is a good modification, the cost of the part detached from agent a_i's piece is not less than the cost of detached parts from other agents' pieces in her perspective. Therefore, agent a_i does not envy a_{k+1} if we detach $e^{a_1}_{k+1}$ from the snapshots in the working set and give the piece p to a_{k+1}.

Here an important observation is that, while we are using modifying the reserved snapshots, an agent might think that the cost of the detached part is very significant. Recall that mask of every $e^{a_1}_{k+1}$ is the same, so by partially detaching some of the $e^{a_1}_{k+1}$ parts, at least one agent thinks that the cost of the detached part is very insignificant since we can find a agent who thinks that cost of every detached part is very insignificant. Therefore we have found a high discrepancy in agents' valuation of detached parts in comparison to R. We give this discrepant part to Discrepancy Protocol to allocated the whole chore.

Due to what we said, if we have a good set of reserved snapshots for every agent in A' with properties mentioned above, we can modify the reserved snapshots such that no agent in A' envy a_{k+1}. To this end, we reserve some of the snapshots every time the protocol finds a cycle and exchanges pieces. We also reserve some snapshots at the beginning of the protocol. In particular, every time we exchange pieces and give a new piece to an agent a_i, we ask other agents to reserve some snapshots. Specifically, if we give a new piece to agent a_i, we ask every other agent a_j which had a trim on this piece including the a_i herself, to reserve for every part e on this piece, the half of the snapshots which cost of e is the most. Suppose that this piece was owned by agent a_l at the beginning of the protocol, we use $Save^{a_l,z}_{a_j,a_i}$ to denote the reserved snapshots that agent a_j reserved them considering the cost of z^{th} part of the piece originally owned by a_l which is $e^{a_1}_z$ when this piece is reassigned to a_i. At the beginning of the protocol, we ask each agent to reserve half of the snapshots according to the cost of each part of each piece. We use $Save^{a_i,z}_{a_j,a_i}$ to denote the snapshots that agent a_j reserved at the beginning of the protocol according z^{th} part of agent a_i's piece.

Again suppose that we have relabeled the agents and we want to detach $e^{a_1}_{k+1}$ from the snapshots in the working set. It is easy to see that for an agent a_i, the set of snapshots $Save^{a_1,k+1}_{a_i,a_1}, \cdots, Save^{a_1,k+1}_{a_i,a_k}$ have the all properties mentioned above. The size of each of them is at least $|W|$, in $Save^{a_1,k+1}_{a_i,a_j}$ the agent a_j has the piece that we want to detach some part of it, and the cost of $e^{a_1}_{k+1}$ is greater than the cost of same part in the working set's snapshots.

We modify the reserved snapshots as mentioned before, to preserve envy-freeness while we are exchanging the pieces. The Permutation Protocol keeps doing the same thing until it finds a piece that had less than $n-1$ trims on it and all its trims are detached. Therefore, it finds set of agents who have a significant bonus over others and return this set of agents.

4.7 The Discrepancy Protocol

The Discrepancy Protocol is responsible for dominating a set of agent to others whenever we find an unallocated piece such that there is a high discrepancy in the agents' evaluation of this piece.

Suppose that we have detached a piece that is very significant for a set of agents B and very insignificant for others. Since the modifications in the Permutation Protocol were very insignificant for all the agents, for each agent a_i in B we have:

$$C_{a_i}(e) \geq C_{a_i}(R) \times 2^{n+1}$$

and for every other agent in $C = A \setminus B$ we have:

$$C_{a_i}(R) \geq C_{a_i}(e) \times 2^{n+1}$$

So e costs at least 2^{n+1} times more than R for agents in B. We call Near Exact Protocol and ask agents in B to divide the e into $2^{|C|+1}$ pieces with $\epsilon = 1/(2^{|C|+2})$, therefore each piece costs at least $C_{a_i}(e)/(2^{|C|+2}) \geq 1/2^{n+1}$ where a_i is an agent in B. By calling the Oblige Protocol for these pieces and agents C, we get a partial envy-free allocation of e such that each agent in C gets at least one intact piece. Therefore, each agent a_i in B thinks that every other agent C has got a piece that costs at least $\frac{C_{a_i}(e)}{2^{n+1}} \geq C_{a_i}(R)$. Hence, even if an agent in B gets all the R, she will not envy any agent in C. However, agents in C are not happy with this assignment, since we are given a part of the chore to them, without giving anything to agents in B. Similarly R costs at least 2^{n+1} times more than e for agents in C. Again we do the same thing and run Near Exact Protocol to divide R and partially assign it to agents in B. Finally, we recursively allocated the remaining pieces of R between agents in B and remaining pieces of e between agents in C. Due to what we said, this assignment would preserve envy-freeness and allocate the whole chore.

4.8 Near-Exact Protocol

In this section we provide a protocol which partitions a chore R into m pieces that are almost equal for every agent. [88] first provided an alternative envy-free cake cutting protocol using a partitioning of a cake into m pieces that are almost equally valuable for every agent. Here we prove that a similar approach works for negative valuations and provide the full proof for completeness. In particular we prove the following lemma.

Lemma 4.8.1 *Given a chore R, an integer m, and a real number $\epsilon > 0$, there exists a bounded protocol that partitions R into pieces $P_1, \ldots, P_m$, such that for every agent a, and $1 \leq i \leq m$*

$$|C_a(P_i) - C_a(C)/m| \leq \epsilon C_a(C),$$

and moreover, for agent a_n, $C_{a_n}(P_i) = C_{a_n}(C)/m$.

Proof. The proof is by an induction on the number of agents n. The statement is trivial for the case of $n = 1$. Let $\epsilon' \in \mathbb{R}$ and $t \in \mathbb{N}$ be two number to be defined later. Due to induction hypothesis, there exists a bounded protocol that partitions R into mt pieces $P'_1, \ldots, P'_{mt}$, such that for all $m-1$ first agents, $|C_a(P'_i) - C_a(R)/mt| \leq \epsilon' C_a(R)$. Now without loss of generality assume that P'_i are in decreasing order according to a_n's cost function, or in other words for $i < j$, $C_{a_n}(P'_i) \geq C_{a_n}(P'_j)$. Now let $Q = P'_1 \cup \ldots \cup P'_m$, and $Q_i = P'_{m+i} \cup P'_{2m+i} \cup \ldots \cup P'_{(t-1)m+i}$ for each $1 \leq i \leq m$. For a_n, Q_1 and Q_m are the largest and the smallest pieces, respectively. Since $C_{a_n}(Q_1) - C_{a_n}(Q_m) \leq C_{a_n}(P'_{m+1})$, a_n can divide Q among $Q_1, \ldots, Q_m$ to make them equal and obtain pieces $P_1, \ldots, P_m$. Now for every agent a and piece P,

$$(t-1)(\frac{C_a(R)}{mt} - \epsilon' C_a(R)) \leq C_a(P) \leq (t-1+m)(\frac{C_a(R)}{mt} + \epsilon' C_a(R)).$$

There we can choose ϵ' small enough and t large enough to have

$$|C_a(P_i) - C_a(R)/m| \leq \epsilon C_a(R).$$

□

4.9 The Oblige Protocol

The Oblige Protocol is called by the Discrepancy Protocol to get a partial envy-free allocation such that every agent gets at least one complete piece. The Discrepancy Protocol uses this property to make a set of agents dominant to others.

The Oblige Protocol gets set of n agents and a partitioning of the chore into 2^{n+1} pieces and returns a partial envy-free allocation such that for every agent at least one piece is completely assigned to her. It asks agents $a_1, a_2, \cdots, a_n$ respectively, asking agent a_i to set aside her 2^{i-1} most costly pieces. Then it asks agents $a_n, a_{n-1}, \cdots, a_1$ respectively, asking agent a_i to return some part of her set aside pieces to create a $2^{i-1} + 1$-way tie for her smallest pieces. Finally, it asks agents $a_1, a_2, \cdots, a_n$ respectively, to choose their smallest piece.

Lemma 4.9.1 *The Protocol returns a partial envy-free allocation such that each agent gets at least one complete piece.*

Proof. The Protocol first asks each agent to reserve some of her largest pieces. After that at least $2^{n+1} - (2^0 + 2^1 + \cdots + 2^{n-1}) \geq 2^n$ pieces will remain. In lines 4 - 5 each agent is asked to create a tie between her smallest pieces using her reservation. When agent a_i is asked to create a tie between her smallest pieces, there will be at least $2^{i-1} + 1$ intact pieces, therefore her $(2^{i-1} + 1)^{th}$ smallest piece has a value less than or equal to value of one of the remaining intact pieces which are not greater than the values of her reserved pieces. Therefore, for each of her 1^{st}, 2^{nd},..., $2^{i-1^{th}}$ smallest pieces, she can return back some part of one of her reserved pieces to equalize them with the value of $(2^{i-1} + 1)^{th}$ smallest piece. Next, each agent a_i is asked to take one of her smallest pieces. Since after agent a_i equalized her smallest pieces at most 2^{i-1} pieces are allocated or augmented, at least one of the smallest pieces are available and she can take it. $\square$

Algorithm 9: Main Protocol

```
Data: List of agents $A = \{a_1, a_2, ..., a_n\}$ and chore $R$
1  if $n = 1$ then
2      Give the whole chore to the agent $a_1$ ;
3      return the allocation;
4  end
5  else if $n = 2$ then
6      Run divide and choose procedure for agents $a_1$ and $a_2$ and chore $R$ ;
7      return the allocation;
8  end
9  else
10     for $i = 1...IS_n \times n^{n^{n^n}}$ do
11         Run Core Protocol($a_1, A, R$) to create snapshot $s_i$ and update the remaining
           chore;
12     end
13     for $i = 1...IS_n \times n^{n^{n^n}}$ do
14         for every pair of agent $a, b$ such that $Adv_{a,b}^{s_i}$ is not significant do
15             Ask agent $a$ to place a trim on $s_i^b$ to make it equal to $s_i^a$ ;
16         end
17     end
18     while there are is a snapshot $s_i$ and pair of agents $a$ and $b$ such that
       $C_a(R)(\frac{1}{2^{2n}}) \leq Adv_{a,b}^{s_i} \leq 2^{2n}C_a(R)$ or
19     $C_a(R)(\frac{1}{2^{2n}}) \leq C_a(e_j^{s_i,b}) \leq 2^{2n}C_a(R)$ for some j do
20         Run Core Protocol($a, A, R$);
21         if an agent $c$ got a significant advantage over agent $d$ in a snapshot $s'$ then
22             Remove trim line of agent $c$ from $s'^d$ ;
23         end
24     end
25     if there is a set of agents $B \subset A$ which every agents in $B$ has significant every
       other agent in $A \setminus B$ then
26         for each agent $a_i$ do
27             Call Core Protocol($a_i, A, R$) ;
28         end
29         Call Main($B$,$R$) ;
30         return the allocation ;
31     end
32     Find the set $S$ with the size of $IS_n$ of isomorphism snapshots;
33     Run the Permutation Protocol($C$, $A$, $R$);
34     Let $B$ be set of agents returned by the Permutation Protocol;
35     for each agent $a_i$ do
36         Call Core Protocol($a_i, A, R$) ;
37     end
38     Call Main($B$,$R$) ;
39     return the allocation ;
40 end
```

Algorithm 10: Existential Core Protocol

```
Data: Agent set $A = \langle a_1, a_2, \ldots, a_n \rangle$, specified cutter $a_{cutter} \in A$, and unallocated
      chore $R$
1  Specified cutter $a_{cutter}$ divides the chore into $n$ equal pieces according to her own
   perspective;
2  Define $p_1, p_2, \ldots, p_n$ the pieces that we have after the division of $a_{cutter}$;
3  for each permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the agents do
4      for each permutation $\langle p_{a'_1}, p_{a'_2}, \ldots, p_{a'_n} \rangle$ of the pieces do
5          Ask $a'_1$ to make a trim on the rightmost side of the $p_{a'_1}$;
6          Allocate $p_{a'_1}$ to $a'_1$;
7          for $i$ from 2 to $n$ do
8              Ask $a'_i$ to trim $p_{a'_i}$ to equalize it with her most preferred piece among
               the first $i - 1$ allocated pieces (we consider the cost value of each piece
               from its leftmost side to its trim.);
9              Allocate $p_{a'_i}$, from its leftmost side to its trim, to $a'_i$;
10         end
11         if none of the agents envies to another agent then
12             return the envy-free partial allocation (at least one of the pieces has
               been completely allocated) and the unallocated chore;
13         end
14         Ignore the previous trims and deallocate the allocated pieces;
15     end
16 end
```

Algorithm 11: Core Protocol

```
Data: Agent set $A = \langle a_1, a_2, \ldots, a_n \rangle$, specified cutter $a_{cutter} \in A$, and unallocated
      chore $R$
1  Specified cutter $a_{cutter}$ divides the chore into $n$ equal pieces according to her own
   perspective;
2  Define $p_1, p_2, \ldots, p_n$ the pieces that we have after the division of $a_{cutter}$;
3  Define main trims $t_1, t_2, \ldots, t_n$ for pieces $p_1, p_2, \ldots, p_n$ respectively where they are
   initially on the rightmost side of the pieces;
4  Run Separated Chore Core Protocol on all the agents and all the pieces with their
   main trims. The call gives an envy-free partial allocation (at least the main trim of
   one of the pieces is not changed);
5  return envy-free partial allocation (at least one of the pieces has been completely
   allocated) and the unallocated chore;
```

Algorithm 12: Separated Chore Core Protocol

```
Data: A chore with $n$ pieces $\langle p_1, p_2, \ldots, p_n\rangle$ with their main trims $\langle t_1, t_2, \ldots, t_n\rangle$
      consecutively and a set of agents $A = \{a_1, a_2, \ldots, a_n\}$
1  Remember the initial main trims of the pieces during this call;
2  for $i = 1$ to $n$ do
3      Agent $a_i$ chooses piece $p \in \{p_1, \ldots, p_n\}$ which is the most preferred piece for
       her among all pieces;
4      if $p$ is not allocated to agents $a_1, a_2, \ldots, a_{i-1}$ then
5          Allocate $p$ to agent $a_i$;
6      end
7      else
8          Suppose that $a_i$ chooses a piece which has been allocated to $a_j$;
9          if the main trim of $p$ is not changed then
10             Run Allocation Extender Protocol for all the pieces with their main
               trims, the first $i$ agents with their current allocation, the popular piece $p$,
               and its fan agents $a_i$ and $a_j$. The call returns a neat allocation of pieces
               to agents (one piece each agent) without changing the main trim of $p$;
11         end
12         else
13             Run Core Match Refiner Protocol for all the pieces with their main
               trims, the first $i - 1$ agents with their current allocation, and the popular
               piece $p$. The call returns a refined neat allocation and a flag (In the case
               the flag is true, it returns the new popular piece $q$ with agent $a$ as its
               happy fan and agent $b$ as its sad fan);
14             Allocate $p$ to $a_i$;
15             if flag = true then
16                 Run Allocation Extender Protocol for all the pieces with their main
                   trims, the first $i$ agents with their current allocation, the popular
                   piece $q$, and its fan agents $a$ and $b$. The call returns a neat allocation
                   of pieces to agents (one piece each agent) without changing the
                   main trim of $q$;
17             end
18         end
19     end
20     Run Best Piece Equalizer Protocol on all $n$ pieces with their main trims, all the
       first $i$ agents, and the current allocation. The call gives a neat allocation of
       pieces to the called agents without any bad subset of agents;
21 end
22 return envy-free partial allocation (with at least one intact piece) and the
   unallocated chore;
```

Algorithm 13: Allocation Extender Protocol

```
Data: A chore with $m$ pieces $\langle p_1, p_2, \ldots, p_m\rangle$ with their main trims $\langle t_1, t_2, \ldots, t_m\rangle$
      consecutively, a set of agents $A = \{a_1, a_2, \ldots, a_n\}$ ($n \leq m$) with a neat
      allocation of pieces to agents, one specific popular piece
      $p \in \{p_1, p_2, \ldots, p_m\}$, and its two specific happy fan agent $a \in A$ as well as
      sad fan agent $b \in A$
1  Remember the initial main trims of the pieces during this call;
2  for each piece $q \in \{p_1, \ldots, p_m\}$ do
3      Ask agents $a$ and $b$ to trim $q$ equal to $p$;
4      Set the main trim of piece $q$ as the rightmost trim among the trims that agents $a$
       and $b$ made on $q$;
5  end
6  Deallocate the allocated pieces but remember the allocation as the original
   mapping;
7  Run the SubCore Protocol on all $m$ pieces except $p$ with their main trims and all
   agents except $a$ and $b$ with their original mapping. The call gives a neat allocation
   of pieces to the called agents;
8  After the allocation, at least one piece $q$ among pieces $\{p_1, \ldots, p_m\}$ except $p$ is not
   allocated;
9  if the main trim of $q$ is made by agent $a$ then
10     Allocate $q$ to $a$, and $p$ to $b$;
11 end
12 else
13     Allocate $q$ to $b$, and $p$ to $a$;
14 end
15 return neat allocation of pieces to the agents (one piece each agent), without
   changing the main trim of $p$, and the unallocated chore;
```

Algorithm 14: SubCore Protocol

```
Data: A chore with $m$ pieces $\langle p_1, p_2, \ldots, p_m\rangle$ with their main trims $\langle t_1, t_2, \ldots, t_m\rangle$
      consecutively and a set of agents $A = \{a_1, a_2, \ldots, a_n\}$ with an original
      mapping of agents to pieces $\{p_{a_1}, p_{a_2}, \ldots, p_{a_n}\}$ consecutively
1  Define a set $P$ of pieces which is an empty set initially;
2  for $i = 1$ to $n$ do
3      Agent $a_i$ chooses piece $p$ which is the most preferred piece for her among the
       pieces;
4      if $p$ has not been allocated to agents $a_1, a_2, \ldots, a_{i-1}$ then
5          Allocate $p$ to agent $a_i$ tentatively and add $p$ to set $P$;
6      end
7      else
8          Define a representative agent for each piece in $P$ (not assigned yet);
9          for $j = 1$ to $i$ do
10             Find the cost value of each piece outside of $P$ for $a_j$. We define $c_j$ as
               the minimum cost value among these values;
11             for each piece $q \in P$ do
12                 $a_j$ makes a trim on $q$ in a way that equalize it to $c_j$. If the cost value
                   of the piece from its leftmost side to its original mapping trim was
                   not more than $c_j$, $a_j$ makes a trim on the original mapping trim of
                   the piece;
13                 if $a_j$ makes the rightmost trim on $q$ then
14                     Set $a_j$ as the representative of $q$ if she makes the first rightmost
                       trim on it, or $q = p_{a_j}$ holds;
15                 end
16             end
17         end
18         Define set $W$ which initially contains the representative agents of the
           pieces in $P$;
19         while $|W| \leq |P|$ do
20             Ignore the previous trims of agents in $W$ and deallocate the allocated
               pieces;
21             Define set $W' = \{a_1, \ldots, a_i\} \setminus W$;
22             Update the main trim of pieces in $P$ by agents in $W'$ (check Subsection
               4.5.4 for more details);
23             Find the modified original mapping of $W$ to $P$ (check Subsection 4.5.4
               for more details);
24             Run SubCore Protocol on all pieces in $P$ with their main trims, the
               agents in $W$, and the modified original mapping of $W$ as the original
               mapping. The call gives an allocation of pieces to agents in $W$;
25             if $|W| = |P|$ then
26                 Break;
27             end
28             Take an arbitrary piece $q$ from unallocated pieces of $P$ such that
               $a \in W'$ is the agent with the lowest index among the agents who have
               the rightmost trim on $q$;
29             Allocate $q$ to $a$ and add $a$ to $W$;
30         end
31         Let $a$ be the only agent among $a_1, \ldots, a_i$ who is not in $W$;
```

Algorithm 15: Best Piece Equalizer Protocol

```
Data: A chore with $m$ pieces $\langle p_1, p_2, \dots, p_m \rangle$ with their main trims $\langle t_1, t_2, \dots, t_m \rangle$
      consecutively and a set of $n$ agents $A = \{a_1, a_2, \dots, a_n\}$ ($n \leq m$) with an
      allocation of the pieces $\{p_{a_1}, p_{a_2}, \dots, p_{a_n}\}$ to the agents $\{a_1, a_2, \dots, a_n\}$
      consecutively
1  while there exists a bad subset $S \subseteq A$ of the agents do
2      Define set $P$ as the set of pieces that we have allocated to $S$;
3      while $S$ is a bad set do
4          for each agent $a \in S$ do
5              for each piece $p \in P$ do
6                  Ask agent $a$ to make a trim on $p$ to equalize it with her most
                   preferred piece out of $P$ (If the cost value of $p$ from its leftmost side
                   to its rightmost side was less than the cost value of her most
                   preferred piece out of $P$, she makes a trim on the rightmost side of
                   $p$.);
7              end
8          end
9          Define an equalizer trim for each piece in $P$ (We have not set them yet);
10         for each piece $p \in P$ do
11             Set the leftmost trim on $p$ which is righter than its main trim as the
               equalizer trim of $p$;
12         end
13         Set the current allocation of pieces to agents as the old allocation;
14         Run Separated Chore Core Protocol on all the agents in $S$ and all the pieces
           in $P$ with their equalizer trims as their main trims. The call gives an
           envy-free partial allocation of the pieces to agents, which we call it the new
           allocation, and updates the main trims (At least the main trim of one of the
           pieces is not changed);
15         Run Monotonicity Saver Protocol for the agents in $S$, the pieces in $P$, and
           the old as well as new allocations of pieces to $S$. The call gives an
           envy-free partial allocation of the called pieces to the called agents, and it
           updates the main trims to keep the protocol monotone.
16     end
17 end
18 return a neat allocation (without any bad subset of agents) of pieces to $A$ (one
   piece each agent) with their updated main trims;
```

Algorithm 16: Monotonicity Saver Protocol

```
Data: A chore with $n$ pieces $\langle p_1, p_2, \ldots, p_n \rangle$, a set of $n$ agents
      $A = \{a_1, a_2, \ldots, a_n\}$, an old allocation of the pieces $\{p_{a_1}, p_{a_2}, \ldots, p_{a_n}\}$ to
      the agents $\{a_1, a_2, \ldots, a_n\}$ consecutively, a new allocation of pieces to
      agents, and the main trims of the allocations
1 Let $P$ be the set of pieces whose the main trim in the new allocation is lefter than
  its main trim in the old allocation;
2 Let $S$ be the set of agents who owns the pieces in $P$;
3 Deallocate the pieces that we have allocated to $S$ in the new allocation;
4 for each agent $a_i \in S$ do
5     Change the main trim of $p_{a_i}$ to its main trim in the old allocation;
6     Allocate $p_{a_i}$ to $a_i$;
7 end
8 return the updated new allocation which is an envy-free partial allocation of pieces
  to $A$ (one piece each agent);
```

Algorithm 17: Core Match Refiner Protocol

```
Data: A chore with $m$ pieces $\langle p_1, p_2, \dots, p_m \rangle$, a set of $n$ agents
      $A = \{a_1, a_2, \dots, a_n\}$ $(n \leq m)$ with a neat allocation of pieces to agents
      (without any bad subset of agents), and a specific popular piece
      $p \in \{p_1, p_2, \dots, p_m\}$
1  Build the Core Match Refiner Graph $G$;
2  Define a directed path $P$ which is initially $P = \langle v_p \rangle$ where $v_p$ is the vertex of piece
   $p$;
3  while the piece of the last vertex of $P$ is not an unallocated or an intact piece do
4      Find a vertex $v_r$ out of $P$ such that there exists an edge from $v_q$ one of the
       vertices of $P$ to it. Let $v_r$ and $v_q$ be the vertices of pieces $r$ and $q$ consecutively;
5      Remove all the vertices after $v_q$ from the path;
6      Add $v_r$ to the end of the path;
7  end
8  Let $P = \langle v_1, v_2, \dots, v_k \rangle$ be the final path;
9  Define a flag which is false if the piece of vertex $v_k$ be an unallocated piece, and
   true otherwise;
10 if flag = true then
11     Let agents $a$ and $b$ be the owners of the pieces of vertices $v_{k-1}$ and $v_k$
       consecutively;
12     Let piece $q$ be the piece of vertex $v_k$. Define $q$ as a popular piece with agents $a$
       and $b$ as its fans;
13     Deallocate the piece of vertex $v_k$ from agent $b$;
14 end
15 for $i$ from $k - 1$ to $1$ do
16     Deallocate the piece of vertex $v_i$ from its owner agent $c$;
17     Allocate the piece of vertex $v_{i+1}$ to agent $c$;
18 end
19 return the refined neat allocation and the flag (In the case the flag is true, return the
   popular piece $q$, its happy fan $a$, and sad fan $b$) ;
```

Algorithm 18: Permutation Protocol

```
   Data: List of agents $A = \{a_1, a_2, ..., a_n\}$ and $IS_n$ isomorphic snapshots
         $S = \{s_1, s_2, ..., s_{IS_n}\}$
1  Set boolean flag $done$ equal to $false$ ;
2  for $i = 1..n$ do
3      for $i' = 1..n$ do
4          for $j = 1...l_{a'_i}$ do
5              Ask agent $a_i$ to reserve $\frac{1}{2}$ of the snapshots in $W$ which the have the
               most cost for $e_j^{a_{i'}}$ and add them to $Save_{a_i,a_{i'}}^{a'_i,j}$ ;
6          end
7      end
8  end
9  while $done = false$ do
10     Create an empty graph $G$ ;
11     for each active piece $p_a$ do
12         Add a directed edge from $a$ to the agent who has the next rightmost trim on
           $p_a$ ;
13     end
14     for each inactive piece $p_b$ do
15         Add a directed edge from $b$ to all other agents ;
16     end
17     Let $C = b_1, b_2, \cdots, b_{|C|}$ be the cycle with at least one active node ;
18     for each $b_i$ in $C$ do
19         Let $c$ the next agent in the cycle ;
20         if $b_i$ is an active node then
21             Relabel the agents in order that they had a trim line on $p_{b_i}$, so that agent
               $a_1$ was the first agent who had this piece, agent $a_2$ detached a first part
               and so on.
22             Lets suppose that trim line of agent $a_k$ is already detached and agent $a_k$
               is the owner of this piece and we want to detach from the trim line of
               agent $a_{k+1}$ and give the piece to him.
23             for $i' = 1..k$ do
24                 Take $|W|$ snapshots from each of
                   $Save_{a_{i'},a_1}^{a_1,k+1}, Save_{a_{i'},a_2}^{a_1,l+1}, \cdots, Save_{a_{i'},a_k}^{a_1,l+1}$ and piecewise merge them
                   to get a snapshot $s'$;
25                 Add virtual pieces to make this allocation envy-free ;
26                 for $j = 1..l$ do
27                     Cut $s'^{a_j}$ from the trim of the agent $a_{k+1}$;
28                 end
29                 Run Sub Core Protocol with set of pieces $s'^{a_1}, \cdots s'^{a_k}$ and set of
                   agents $a_1, \cdots a_k$ with their original mapping of these pieces;
30                 if detached parts costs significant for at least one agent then
31                     Let $X$ be the union of the detached parts ;
32                     return Discrepancy Protocol($X$,$R$) ;
33                 end
34             end
35             Detach the right most trim line from $p_{b_i}$ from all of the snapshots in
               Working Set ;
```

Algorithm 19: Discrepancy Protocol

```
   Data: List of agents $A = \{a_1, a_2, ..., a_n\}$, a discrepant piece $e$ and residual chore $R$
1  Let $B$ be set of agents who thinks $p$ costs very significant.
   $B = \{\forall a_i : C_{a_i}(e) \geq C_{a_i}(R) \times 2^n\}$ ;
2  Let $B$ be other agents. $C = A \setminus B$ ;
3  Let $P_C$ be the output of Near Exact($B$,$e$,$2^{|C|+1}$,$1/(2^{|C|+2})$) ;
4  Let $P_B$ be the output of Near Exact($C$,$R$,$2^{|B|+1}$,$1/(2^{|B|+2})$) ;
5  Call Oblige Protocol($C$,$P_C$) ;
6  Call Oblige Protocol($B$,$P_B$) ;
7  Let $R_B$ be the unallocated parts of $P_B$ ;
8  Run Main Protocol($B$,$R_B$) ;
9  Let $R_C$ be the unallocated parts of $P_C$ ;
10 Run Main Protocol($C$,$R_C$) ;
11 return $B$ ;
12 (Since the whole chore is already allocated, every agent dominates the others.)
```

Algorithm 20: Oblige Protocol

```
   Data: List of agents $A = \{a_1, a_2, ..., a_n\}$ and partition of chore into $2^{n+1}$ pieces
         $p_1, p_2, ..., p_{2^{n+1}}$.
1  for $i = 1...n$ do
2      Ask agent $a_i$ to set aside her $2^{i-1}$ largest pieces. ;
3  end
4  for $i = n...1$ do
5      Ask agent $a_i$ to return part of her reserved pieces to the remaining pieces to
       create a $2^{i-1} + 1$-way tie for her smallest pieces ;
6  end
7  for $i = 1...n$ do
8      Ask agent $a_i$ to choose her smallest piece in the remaining pieces. Each agent
       have to take a piece she augmented if one is available;
9      (Break ties in lexicographic order.)
10 end
11 return partial envy-free allocation and remaining of the chore ;
```

Chapter 5: Fair Allocation of Indivisible Goods to Asymmetric Agents

5.1 Introduction

In this work, we conduct a study of *fairly* allocating *indivisible goods* among n agents with unequal claims on the goods. Fair allocation is a very fundamental problem that has received attention in both Computer Science and Economics. This problem dates back to 1948 when [16] introduced the *cake cutting* problem as follows: given n agents with different valuation functions for a cake, is it possible to divide the cake between them in such a way that every agent receives a piece whose value to him is at least $1/n$ of the whole cake? Steinhaus answered this question in the affirmative by proposing a simple and elegant algorithm which is called *moving knife*. Although this problem admits a straightforward solution, several ramifications of the cake cutting problem have been studied since then, many of which have not been settled after decades [68, 69, 89, 85, 81, 90, 91, 92, 93, 94, 95]. For instance, a natural generalization of the problem in which we discriminate the agents based on their entitlements is still open. In this problem, every agent claims an entitlement e_i to the cake such that $\sum e_i = 1$, and the goal is to cut the cake into disproportional pieces and allocate them to the agents such that every agent a_i's valuation for his piece is at least e_i fraction of his valuation for the entire cake. For two agents, Brams *et al.* [96] showed that at least two cuts are necessary to divide the cake between the agents. Furthermore, Robertson *et al.* [97] proposed a modified version of cut and choose method to divide the cake between two agents with portions e_1, e_2, where e_1 and e_2 are real numbers. McAvaney, Robertson, and Web [98] considered the case when the entitlements are rational numbers. They used Ramsey partitions to show that when the entitlements are rational, one can make a proper division via $O(n^3)$ cuts.

Recently, a new line of research is focused on the fair allocation of indivisible goods. In contrast to the conventional cake cutting problem, in this problem instead of a heterogeneous cake, we have a set $\mathcal{M}$ of indivisible goods and we wish to distribute them among n agents. Indeed, due to trivial counterexamples in this setting[1], the previous guarantee, that is every agent should obtain $1/n$ of his valuation for all items from his allocated set, is impossible to deliver. To alleviate this problem, Budish [1] proposed a concept of fairness for the allocation of indivisible goods namely *the maxmin share*. Suppose we ask an agent a_i to divide the items between the agents in a way that *he thinks* is fair to everybody. Of course, agent a_i does not take into account other agents' valuations and only incorporates his valuation function in the allocation. Based on this, we define MMS_i equal to the minimum profit that any agent receives in this allocation, according to agent a_i's valuation function. Obviously, in order to maximize MMS_i, agent a_i chooses an allocation that maximizes the minimum profit of the agents. We call an allocation fair (approximately fair), if every agent a_i receives a set of items that is worth at least MMS_i (a fraction of MMS_i) to him.

It is easy to see that MMS_i is the best possible guarantee that one can hope to obtain in this setting. If all agents have the same valuation function, then at least one of the

[1]For instance if there is only one item, at most one agent has a non-zero profit in any allocation.

agents receives a collection of items that are worth no more than MMS_i to him. A natural question that emerges here is whether a fair allocation with respect to MMS_i's is always possible? Although the experiments are in favor of this conjecture, Kurokawa, Procaccia, and Wang [2] (EC'14) refuted this by an elegant and delicate counterexample. They show such a fair allocation is impossible in some cases, even when the number of agents is limited to 3. On the positive side however, they show an approximately fair allocation can be guaranteed. More precisely, they show that there always exists an allocation in which every agent's profit is at least $2/3\mathsf{MMS}_i$. Such an allocation is called a $2/3$-MMS allocation. Amanatidis, Markakis, Nikzad, and Saberi [12] later provided a proof for the existence of an MMS allocation for the case, when there are large enough items and the value of each agent for every items is drawn independently from a uniform distribution. A generalized form of this result was later proposed by Kurokawa *et al*. [23] for arbitrary distributions. In a recent work, Ghodsi *et al*. [3] provided a proof for existence of a $3/4$-MMS allocation.

Although it is natural to assume the agents have equal entitlements on the items, in most real-world applications, agents have unequal entitlements on the goods. For instance, in various religions, cultures, and regulations, the distribution of the inherited wealth is often unequal. Furthermore, the division of mineral resources of a land or international waters between the neighboring countries is often made unequally based on the geographic, economic, and political status of the countries.

For fairly allocating indivisible items to agents with different entitlements, two procedures are proposed in [68]. The first one is based on *Knaster's procedure of sealed bids*. In this method, we have an auction for selling each item. Therefore, for using it all the agents should have an adequate reserve of money which is the main issue of the procedure. The second procedure mentioned in [68] is based on *method of markers* developed by William F. Lucas which is spiritually similar to the moving knife procedure. In this method, first we line up the items, and then the agents place some markers for dividing the items. This method suffers from high dependency of its final allocation to the order of the items in the line.

Agent duplication is another idea to deal with unequal entitlements. More precisely, when all of the entitlements are fractional numbers, we can duplicate each agent a_i to some agents with similar valuation functions to a_i. The goal of this duplication is to reduce the problem to the case of equal entitlements. After the allocation, every agent a_i owns all of the allocated items to her duplicated agents. For instance, assume that we have three agents with entitlements $1/2$, $2/5$, and $1/10$, respectively. In this case, we duplicate the first agent to five agents and the second agent to four agents each having an entitlement of $1/10$. This way, we can reduce our problem to the case of equal entitlements. Although agent duplication may be practical when the items are divisible, in the indivisible case, this method does not apply to the indivisible setting. For instance, if the number of the agents is higher than the number of available items, we cannot allocate anything to some agents. Another issue with this method is that it works only for fractional entitlements.

In this chapter, we study fair allocation of indivisible items with different entitlements using a model which resolves the mentioned issues. Our fairness criterion mimics the general idea of Budish for defining maxmin shares. Similar to Budish's proposal, in order to define a maxmin share for an agent a_i, we ask the following question: how much benefit does agent a_i expect to receive from a fair allocation, if we were to divide the goods *only based on his valuation function*? If agent a_i expects to receive a profit of p from the allocation, then he should also recognize a minimum profit of $p \cdot e_j/e_i$ for any other agent a_j, so that his own profit per entitlement is a lower bound for all agents. Therefore, a fair answer to this question is the maximum value of p for which there exists an allocation such that agent a_i's profit-per-entitlement can be guaranteed to all other agents (according to his own valuation function). We define the maxmin shares of the agents based on this intuition.

Recall that we denote the number of agents with n and the entitlement of every agent a_i with e_i. We assume the entitlements always add up to 1. For every agent a_i, we define the weighted maxmin share denote by WMMS_i, to be the highest value of p for which there exists an allocation of the goods to the agents in which every agent a_j receives a profit of at least $p \cdot e_j/e_i$ based on agent a_i's valuation function. Similarly, we call an allocation α-WMMS, if every agent a_i obtains an α fraction of WMMS_i from his allocated goods. Notice that in case $e_i = 1/n$ for all agents, this definition is identical to Budish's definition. Since our model is a generalization of the Budish's model, it is known that a fair allocation is not guaranteed to exist for every scenario. However, whether a $2/3$ approximation or in general a constant approximation WMMS allocation exists remains an open question.

Our main result is in contrast to that of Kurokawa, Procaccia, and Wang. We settle the above question by giving a $1/n$ hardness result for this problem. In other words, we show no algorithm can guarantee any allocation which is better than $1/n$-WMMS in general. We further complement this result by providing a simple algorithm that guarantees a $1/n$-WMMS allocation to all agents. As we show in Section 5.3, this hardness is a direct consequence of unreasonably high valuation of agents with low entitlements for some items.

In contrast to our theoretical results, we show in practice a fair allocation is likely to exist by providing experimental results on real-world data. The source of our experiments is a publicly available collection of bids for eBay goods and services[2]. Note that since those auctions are *truthful*[3], it is the users' best interest to bid their actual valuations for the items and thus the market is transparent. More details about the experiments can be found in Section 5.4. We also support our claim by presenting theoretical analysis for the stochastic variants of the problem in which the valuation of every agent for a good is drawn from a given distribution.

5.2 Our Model

Let $\mathcal{N}$ be a set of n agents, and $\mathcal{M}$ be a set of m items. Each agent a_i has an *additive* valuation function V_i for the items. In addition, every agent a_i has an entitlement to the items, namely e_i. The entitlements add up to 1, i.e., $\sum e_i = 1$.

Since our model is a generalization of maxmin share, we begin with a formal definition of the maxmin shares for equal entitlements, proposed by Budish [1]. In this case, we assume all of the entitlements are equal to $1/n$. Let $\Pi(\mathcal{M})$ be the set of n-partitionings of the items. Define the maxmin share of agent a_i (MMS_i) of player i as

$$\mathsf{MMS}_i = \max_{\langle A_1, A_2, \ldots, A_n \rangle \in \Pi(\mathcal{M})} \min_{j \in [n]} V_i(A_j). \tag{5.1}$$

[2]http://cims.nyu.edu/ munoz/data/

[3]An action is called truthful, if no bidder has any incentive to misrepresent his valuation

One can interpret the maxmin share of an agent as his outcome as a divider in a divide-and-choose procedure against adversaries [1]. Consider a situation that a cautious agent knows his own valuation on the items, but the valuations of other agents are unknown to him. If we ask the agent to run a divide-and-choose procedure, he tries to split the items in a way that the least valuable bundle is as attractive as possible.

When the agents have different entitlements, the above interpretation is no longer valid. The problem is that the agents have different entitlements and this discrepancy must somehow be considered in the divide-and-choose procedure. Thus, we need an interpretation of the maxmin share that takes the entitlements into account.

Let us get back to the case with the equal entitlements. Another way to interpret maxmin share is this: suppose that we ask agent a_i to fairly distribute the items in $\mathcal{M}$ between n agents of $\mathcal{N}$, based on his own valuation function. In an ideal situation (e.g., if the goods are completely divisible), we expect a_i to allocate a share with value $V_i(\mathcal{M})/n$ to every agent. However, since the goods are indivisible, some sort of unfairness is inevitable. For this case, we wish that a_i does his best to retain fairness. MMS_i is in fact, a parameter that reveals how much fairness a_i can guarantee, regarding his valuation function.

Formally, to measure the fairness of an allocation by a_i, define a value F^i_A for any allocation $A = \langle A_1, A_2, \ldots, A_n \rangle$ as

$$F^i_A = \frac{\min_j V_i(A_j)}{V_i(\mathcal{M})/n}.$$

In fact, we wish to make sure a_i reports an allocation A^* such that $F^i_{A^*}$ is as close to 1 as possible. The maxmin share of a_i is therefore defined as

$$\mathsf{MMS}_i = F^i_{A^*}(V_i(\mathcal{M})/n). \tag{5.2}$$

It is easy to observe that Equations (5.1) and (5.2) are equivalent, since the fairest allocation in the absence of different entitlements is an allocation that maximizes value of the minimum bundle:

$$\begin{aligned}\mathsf{MMS}_i &= F^i_{A^*}(V_i(\mathcal{M})/n) \\ &= \frac{\min_j V_i(A^*_j)}{V_i(\mathcal{M})/n}(V_i(\mathcal{M})/n) = \min_j V_i(A^*_j)\end{aligned}$$

Now, consider the case with different entitlements. Let e_i be the entitlement of agent a_i. Similar to the second interpretation for MMS_i, ask agent a_i to fairly distribute the items between the agents, but this time, considers the entitlements. In an ideal situation (e.g., a completely divisible resource), we expect the allocation to be proportional to the entitlements, i.e. a_i allocates a share to agent a_j with value exactly $V_i(\mathcal{M})e_j$ (note that when the entitlements are equal, this value equals to $V_i(\mathcal{M})/n$ for every agent). But again, such an ideal situation is very rare to happen and thus we allow some unfairness. In the same way, define the fairness of an allocation $A = \langle A_1, A_2, \ldots, A_n\rangle$ as

$$F^i_A = \min_j \frac{V_i(A_j)}{V_i(\mathcal{M})e_j} \tag{5.3}$$

Let $A^* = \langle A^*_1, A^*_2, \ldots, A^*_n\rangle$ be an allocation by a_i that maximizes $F^i_{A^*}$. The weighted maxmin share of agent a_i is defined in the same way as MMS_i, that is:

$$\mathsf{WMMS}_i = F^i_{A^*} V_i(\mathcal{M}) e_i = e_i \min_j \frac{V_i(A^*_j)}{e_j}$$

In summery, the value WMMS_i for every agent a_i is defined as follows:

$$\mathsf{WMMS}_i = \max_{\langle A_1, A_2, \ldots, A_n \rangle \in \Pi(\mathcal{M})} \min_{j \in [n]} V_i(A_j) \frac{e_i}{e_j}.$$

For more intuition, consider the following example: Assume that we have two agents a_1, a_2 with $e_1 = 1/3$ and $e_2 = 2/3$. Furthermore, suppose that there are 5 items b_1, b_2, b_3, b_4, b_5 with the following valuations for a_1: $V_1(\{b_1\}) = V_1(\{b_2\}) = V_1(\{b_3\}) = 4, V_1(\{b_4\}) = 3$ and $V_1(\{b_5\}) = 9$. For the allocation $A = \langle \{b_5\}, \{b_1, b_2, b_3, b_4\} \rangle$, we have $F_A = \min(\frac{9}{24 \cdot (1/3)}, \frac{15}{24 \cdot (2/3)})$ which means $F_A = 15/16$. Moreover, for allocation $A' = \langle \{b_1, b_2\}, \{b_3, b_4, b_5\} \rangle$, we have $F_{A'} = \min(\frac{8}{24 \cdot (1/3)}, \frac{16}{24 \cdot (2/3)})$ which means $F_{A'} = 1$. Thus, A' is a fairer allocation than A. In addition, A' is the fairest possible allocation and hence, $\mathsf{WMMS}_1 = 1 \cdot 24 \cdot 1/3 = 8$.

This example also gives an insight about why agent duplication (as introduced in the Introduction) is not a good idea. For this example, if we duplicate agent a_2, we have three agents with the same entitlements. But any partitioning of the items into three bundles, results in a bundle with value at most 7 to a_1.

Finally, an allocation of the items in $\mathcal{M}$ to the agents in $\mathcal{N}$ is said to be $\alpha-\mathsf{WMMS}$, if the total value of the share allocated to each agent a_i is worth at least $\alpha\mathsf{WMMS}_i$ to him.

```
Algorithm 21: $1/n$-WMMS allocation
  Input: $\mathcal{N}$, $\mathcal{M}$, valuation functions $V_1, \dots, V_n$, and entitlements $e_1, \dots, e_n$ (without
         loss of generality, sorted in descending order).
  Output: Allocation $A = A_1, \dots, A_n$.
1 ______________________________________________
     2 $i$ from one to $|\mathcal{M}|$ assign an unassigned item $b_j$ to $a_{i \bmod |\mathcal{N}|}$ where
       $V_{i \bmod |\mathcal{N}|}(\{b_j\})$ is maximum among unassigned items
```

5.3 A Tight $1/n$ Bound on the Optimal Allocation

In spite of the fact that there exists a $2/3$-WMMS guarantee when all the entitlements are equal, in our general setting surprisingly we provide a counterexample which proves that there is no guarantee better than $1/n$-WMMS. We complement this result by showing that a $1/n$-WMMS always exists. Thus, these two theorems make a tight bound for the problem.

The main property of our counterexample is a large gap between the value of items for different agents. We provide a counterexample according to this property in Theorem 5.3.1.

Theorem 5.3.1 *There exists no guarantee better than* $1/n$-WMMS *when the entitlements to the items may differ.*

Theorem 5.3.1 gives a $1/n$-WMMS upper-bound. In Theorem 5.3.2, we show that the provided upper-bound is tight. Algorithm 2 uses a simple greedy procedure, which is spiritually similar to an algorithm in [12], guaranteeing $1/n$-WMMS allocation as follow: sort the agents in descending order of entitlements. Starting from the first agent, ask every person to collect the most valuable item from the remaining set, one by one. Repeat the process until no more item is left.

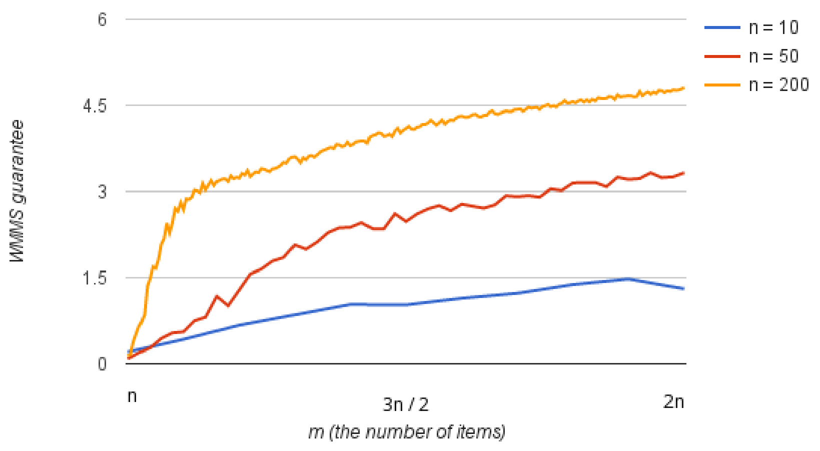


Figure 5.1: The vertical line denotes the ratio of the valuation of the allocated set to the maximin guarantee of the agents. The horizontal line shows the number of items varying from n to $2n$. Blue, red, and yellow poly lines illustrate the performance of our algorithm for $n = 10$, $n = 50$, and $n = 200$ respectively.

Theorem 5.3.2 *Algorithm 2 guarantees a* $1/n$-WMMS *allocation.*

5.4 Empirical Results

As we discussed in Section 5.3, in extreme cases, making a WMMS allocation or even an approximately WMMS allocation is theoretically impossible. However, our counter-example is extremely delicate and thus very unlikely to happen in real-world. Here, we show in practice fair allocations w.h.p exist, especially when the number of items is large.

We draw the valuation of the agents for the goods based on a collection of bids for eBay items publicly available at http://cims.nyu.edu/ munoz/data/. More precisely, for m items, we randomly choose m different categories of goods from the dataset. Moreover, for every agent a_i and item b_j, we set $V_i(\{b_j\})$ to a submitted bid for the corresponding

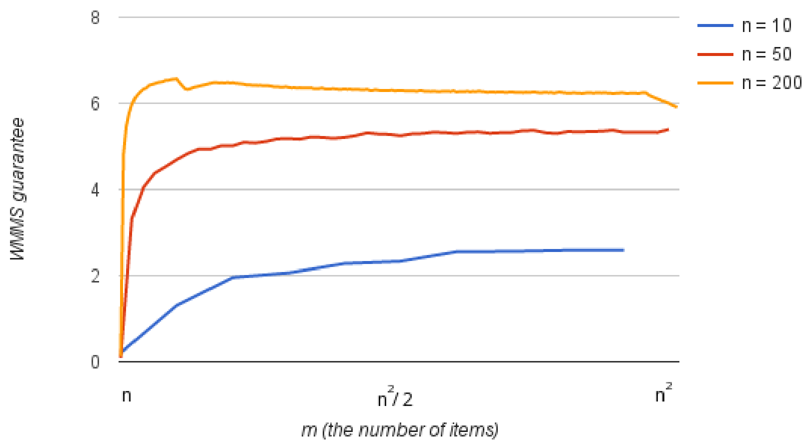


Figure 5.2: The vertical line denotes the ratio of the valuation of the allocated set to the maximin guarantee of the agents. The horizontal line shows the number of items varying from n to n^2. Blue, red, and yellow polylines illustrate the performance of our algorithm for $n = 10$, $n = 50$, and $n = 200$ respectively.

category of item b_j chosen uniformly at random. The bids vary from 0.01 to 113.63 and their mean is 6.57901. Moreover, the expected variance of the bids in every category is 200.513.

For an instance of the problem with n agents and m items, we run the experiments with 1000 different vector of entitlements drawn from the uniform distribution (and scaled up to satisfy $\sum e_i = 1$). For every n and m, we take the minimum WMMS guarantee obtained all 1000 runs, and show it in Figures 5.1 and 5.2. We used heuristic algorithms to compute the maxmin shares and maxmin guarantees. Thus, our results are only *lower bounds* to the actual WMMS guarantees. Nonetheless, the optimal guarantees are very close to the estimated ones.

Figures 5.1 and 5.2 illustrate the result of the runs for $n = 10$, $n = 50$, and $n = 200$ respectively. Figure 5.1 only depicts the WMMS guarantees for $m \in [n, 2n]$ whereas in

Figure 5.2 the number of items varies from n to n^2.

As shown in Figures 5.1 and 5.2, the approximation guarantee improves as we increase the number of items. Moreover, unless m is very close to n, a WMMS allocation exists in our experiments (notice that the guarantee is above 1 when m is considerably larger that n).

5.5 Stochastic Setting

In Section 5.3 we presented a counterexample to show that no allocation better than $1/n$-WMMS can be guaranteed. However, the construction described in the counterexample is very unlikely to happen in the real settings. Here, we show that WMMS allocation exists with high probability when a small randomness is allowed in the setting.

Considering stochastic settings is common in the fair allocation problems since many real-world instances can be modeled with random distributions [1, 23, 12, 55]. The general probabilistic model used in previous works is as follows: every agent a_i has a probability distribution $\mathcal{D}_i$ over $[0, 1]$ and for every item b_j, the value for $V_i(\{b_j\})$ is randomly sampled from $\mathcal{D}_i$. In [12], the existence of an MMS allocation is proved for the special case of $\mathcal{D}_i = U(0, 1)$, where $U(0, 1)$ is the standard uniform distribution with minimum 0 and maximum 1. Kurokawa, Procaccia, and Wang [23] considered the problem for arbitrary random distribution $\mathcal{D}_i$ with the condition that $\mathbb{V}[\mathcal{D}_i] \geq c$ for a positive constant c. A considerable part of the proof for the existence of an MMS allocation in [23] is referred to [55], where the authors proved the existence of an envy-free allocation in the stochastic settings with arbitrary random distributions.

In this section, we consider two different probabilistic models. Our first model is the same as [23], with the exception that we omit the restriction $\mathbb{V}[\mathcal{D}_i] \geq c$. We name this model as *Stochastic Agents* model. In the second model, every item b_i has a probability distribution $\mathcal{D}_i$ and for every agent a_j, value of $V_j(\{b_i\})$ is randomly drawn from $\mathcal{D}_i$. We choose the name *Stochastic Items* for the second model. We believe that *Stochastic Items* model is more realistic since the first model does not make any distinguish between the items. None of the previous works mentioned above considered this model.

We leverage *Hoeffding* inequality to prove the existence of WMMS allocation. Theorem 5.5.1 states the general form of this inequality [99].

Theorem 5.5.1 (General Form of Hoeffding (1963)) *Let $X_1, X_2, \ldots, X_n$ be random variables bounded by the interval $[0,1] : 0 \leq X_i \leq 1$. We define the empirical mean of these variables by $\bar{X} = \frac{1}{n}(X_1 + X_2 + \ldots + X_n)$. Then, the following inequality holds:*

$$\mathbb{P}(|\bar{X} - E(\bar{X})| \geq t) \leq 2e^{-2nt^2} \tag{5.4}$$

Regarding Theorem 5.5.1, let $X = n\bar{X} = \sum_i X_i$ and let $\mu = n \times E(\bar{X})$. By Inequality (5.4), we have:

$$\mathbb{P}(|X - \mu| \geq nt) \leq 2e^{-2nt^2} \tag{5.5}$$

By setting $nt = \delta\mu$, we rewrite Equation (5.5) as:

$$\mathbb{P}(|X - \mu| \geq \delta\mu) \leq 2e^{\frac{-2(\delta\mu)^2}{n}} \tag{5.6}$$

5.5.1 Model I: Stochastic Agents

As mentioned before, in the first model we assume that every agent has a probability distribution $\mathcal{D}_i$ and for every item b_j, the value of $V_i(\{b_j\})$ is randomly sampled from $\mathcal{D}_i$. Furthermore, we suppose $\mu_i = \mathbb{E}(\mathcal{D}_i)$. Throughout this section, we assume that $m \geq n$. This is w.l.o.g , because for the case $m < n$, WMMS_i for every agent a_i equals to zero. For the Stochastic Agents model, we state Theorem 5.5.2.

Theorem 5.5.2 *Consider an instance of the fair allocation problem with unequal entitlements, such that the value of every item for every agent a_i is randomly drawn from distribution $\mathcal{D}_i$. Furthermore, let $s = \min_i s_i$ and $\mu = \min_i \mu_i$. Then, for every $0 < \epsilon < 1$, there exists a value $m' = m'(\frac{1}{e}, \frac{1}{\mu}, \frac{1}{\epsilon})$ (which means m' is a function of $\frac{1}{e}, \frac{1}{\mu}, \frac{1}{\epsilon}$) such that if $m \geq m'$, then almost surely a $(1-\epsilon)$-WMMS allocation exists.*

In the rest of this section, we prove Theorem 5.5.2. Consider the algorithm that allocates $t_i = \lfloor me_i \rfloor$ items to every agent a_i. We know that the value of item b_i for agent a_j is randomly sampled from $\mathcal{D}_i$. From the point of view of the algorithm, it trivially does not matter whether the value of items are sampled after the allocation or before the allocation. Thus, we can suppose that the value of every item is sampled after the allocation of items.

For now, we know that $t_i = \lfloor me_i \rfloor$ number of items are assigned to a_i. Argue that for every $\epsilon > 0$, there exists a value m', such that for every $m \geq m'$, $t_i \geq me_i(1-\epsilon)$. In Lemma 5.5.3 we bound the value of m' in terms of e_i and ϵ.

Lemma 5.5.3 *For every $m > \frac{1}{\epsilon e_i}$, $\lfloor me_i \rfloor \geq me_i(1-\epsilon)$.*

Proof. We know $\lfloor me_i \rfloor \geq me_i - 1 = me_i(1 - \frac{1}{me_i})$. If $m > \frac{1}{\epsilon e_i}$, then $\lfloor me_i \rfloor \geq me_i(1 - \frac{1}{\frac{1}{\epsilon e_i} e_i}) = me_i(1 - \epsilon)$. This, completes the proof. $\square$

For the rest of the proof, suppose that $m > \frac{1}{\epsilon e_i}$. Let X_i be the variable indicating total value of items allocated to a_i. Note that $\mathbb{E}(X_i) = t_i\mu_i$. Regarding Equation (5.6), we have:

$$\mathbb{P}(|X_i - t_i\mu_i| \geq \delta t_i\mu_i) \leq 2e^{\frac{-2(\delta t_i \mu_i)^2}{t_i}}$$

We want to choose δ such that $\mathbb{P}(|X_i - t_i\mu_i| \geq \delta t_i\mu_i) \leq \frac{1}{2mn}$. We have:

$$2e^{\frac{-2(\delta t_i \mu_i)^2}{t_i}} \leq \frac{1}{2mn} \Rightarrow -2(\delta t_i\mu_i)^2 \leq t_i \ln \frac{1}{4mn} \Rightarrow \delta \geq \sqrt{\frac{t_i \ln 4mn}{2(t_i\mu_i)^2}}$$

Regarding the facts that $t_i \geq me_i(1 - \epsilon)$ and $m \geq n$, we have:

$$\sqrt{\frac{t_i \ln 4mn}{2(t_i\mu_i)^2}} \leq \sqrt{\frac{\ln 2 + \ln m}{e_i m {\mu_i}^2 (1 - \epsilon)}}$$

Therefore, it's enough to choose δ such that

$$\delta \geq \sqrt{\frac{\ln 2 + \ln m}{e_i m {\mu_i}^2 (1 - \epsilon)}}. \tag{5.7}$$

Now, let t'_i be the number of items that are not assigned to a_i. Since $t_i + t'_i = m$, regarding the fact that $t_i \geq me_i(1 - \epsilon)$, we have $t'_i \leq m - me_i(1 - \epsilon)$, which means $t'_i \leq m + me_i(\epsilon - 1)$. On the other hand, $t'_i \geq m(1 - e_i)$. Also, let X'_i be the variable indicating total value of the items that are not allocated to a_i. By the same deduction as t_i

for t'_i we have:

$$\mathbb{P}(|X'_i - t'_i\mu_i| \geq \delta' t'_i\mu_i) \leq 2e^{\frac{-2(\delta' t'_i\mu_i)^2}{t'_i}}.$$

Let δ' be the value that $\mathbb{P}(|X_i - t'_i\mu_i| \geq \delta t'_i\mu_i) < \frac{1}{2mn}$. We have:

$$2e^{\frac{-2(\delta' t'_i\mu_i)^2}{t'_i}} \leq \frac{1}{2mn} \Rightarrow (\delta' t'_i\mu_i)^2 \geq \frac{t'_i \ln 4mn}{2}$$

$$\delta' \geq \sqrt{\frac{\ln 4mn}{2t'_i\mu_i^2}} \tag{5.8}$$

Thus, it's enough to choose δ in a way that Inequality (5.8) holds. Regarding the facts that $m > n$ and $t'_i \geq m(1-e_i)$,

$$\sqrt{\frac{\ln 4mn}{2t'_i\mu_i^2}} \leq \sqrt{\frac{\ln 2 + \ln m}{m(1-e_i){\mu_i}^2}}.$$

Therefore, it's enough to choose δ' in a way that

$$\delta' \geq \sqrt{\frac{\ln 2 + \ln m}{m(1-e_i){\mu_i}^2}} \tag{5.9}$$

Now, suppose that both Inequalities (5.7) and (5.9) are held. Considering S_i as the set of items assigned to a_i, with the probability of at least $1 - (\frac{1}{2mn} + \frac{1}{2mn}) = 1 - \frac{1}{mn}$ we have:

$$\frac{V_i(S_i)}{V_i(\mathcal{M})} = \frac{(1-\delta)t_i\mu_i}{(1+\delta)t_i\mu_i + (1+\delta')t'_i\mu_i}$$

It is easy to show that, there always exist an m'_i such that for all $m \geq m'_i$ both Inequalities

(5.7) and (5.9) hold for $\delta = \epsilon$ and $\delta' = \epsilon$. Regarding this, we have:

$$\begin{aligned}
\frac{V_i(S_i)}{V_i(\mathcal{M})} &\geq \frac{(1-\epsilon)t_i\mu_i}{(1+\epsilon)t_i\mu_i + (1+\epsilon)t'_i\mu_i} \\
&= \frac{(1-\epsilon)t_i}{(1+\epsilon)m} \\
&\geq \frac{(1-\epsilon)me_i(1-\epsilon)}{(1+\epsilon)m} \\
&= \frac{(1-\epsilon)^2}{1+\epsilon}e_i = \frac{(1+\epsilon)^2 - 4\epsilon}{1+\epsilon}e_i \\
&= (1+\epsilon)e_i - \frac{4\epsilon}{(1+\epsilon)}e_i \\
&\geq (1+\epsilon)e_i - 4\epsilon e_i = (1-3\epsilon)e_i
\end{aligned}$$

Therefore, with the probability at least $1 - \frac{1}{mn}$ we have:

$$\forall_{a_i \in \mathcal{N}} \qquad \frac{V_i(S_i)}{V_i(\mathcal{M})} \geq (1-3\epsilon)e_i \tag{5.10}$$

Now, suppose that the Inequality 5.10 holds for every agent a_i, with probability at least $1 - \frac{1}{mn}$. Considering all the agents, with the probability at least $(1 - \frac{1}{mn})^n \geq 1 - \frac{n}{nm} = 1 - \frac{1}{m}$, value of $\frac{V_i(S_i)}{V_i(\mathcal{M})}$ for every agent a_i is at least $e_i(1-3\epsilon)$. Regarding the fact that $\mathsf{WMMS}_i \leq e_i V_i(\mathcal{M})$, we have

$$\forall_{a_i \in \mathcal{N}} \qquad V_i(S_i) \geq \mathsf{WMMS}_i(1-3\epsilon)$$

This completes the proof.

5.5.2 Model II: Stochastic Items

As mentioned, in Stochastic Items model, every item b_i has a probability distribution $\mathcal{D}_i$ and the value of every agent a_j for item b_i is randomly chosen from $\mathcal{D}_i$. For this model, we prove Theorem 5.5.4. The theorem states that for large enough m, almost surely a $(1-\epsilon)$-WMMS allocation exists.

Theorem 5.5.4 *Suppose that for every agent a_i, $\mathbb{E}(\mathcal{D}_i) > c$ for a non-negative constant c. Then for all $0 < \epsilon < 1$, there exists $m' = m'(c, \epsilon, n, e_1, ..., e_n)$ such that if $m \geq m'$, then, almost surely, $(1-\epsilon)$-WMMS allocation exists.*

In the rest of the section, we prove Theorem 5.5.4. First, in Lemma 5.5.6 we prove the existence of an allocation that assigns to every agent a_i, a set of items with value at least $\mathsf{WMMS}_i - M_i$, where $M_i = \max_j(V_i(\{b_j\}))$. The idea to prove this fact is inspired by [70]. Argue that we can formulate the allocation problem with unequal entitlements as the following Integer program:

$$\begin{aligned}
&\sum_{b_j \in \mathcal{M}} V_i(\{b_j\}) \cdot f_{i,j} \geq V(\mathcal{M}) \cdot e_i \quad && \forall_{a_i \in \mathcal{N}} \\
&\sum_{a_i \in \mathcal{N}} f_{i,j} = 1 && \forall_{b_j \in \mathcal{M}} \\
&f_{i,j} \in \{0,1\}
\end{aligned} \tag{5.11}$$

In IP(5.11), variable $f_{i,j}$ determines whether b_j is assigned to agent a_i or not. Considering the fact that $V_i(\mathcal{M}) \cdot e_i$ is a trivial upper bound on WMMS_i, any solution to IP5.11 is a feasible solution to the assignment problem with unequal entitlements. By relaxing the second and the third condition, we can convert IP5.11 to LP5.12:

$$
\begin{aligned}
&\sum_{b_j \in \mathcal{M}} V_i(\{b_j\}) \cdot f_{i,j} \geq V(\mathcal{M}) \cdot e_i && \forall_{a_i \in \mathcal{N}} \\
&\sum_{a_i \in \mathcal{N}} f_{i,j} \leq 1 && \forall_{b_j \in \mathcal{M}} \\
&f_{i,j} \geq 0
\end{aligned}
\tag{5.12}
$$

For every feasible solution $\mathcal{A}$ to LP5.12, we construct the bipartite graph $G_{\mathcal{A}}\langle I, J, E\rangle$ where $I = \{1, 2, ..., n\}$ and $J = \{1, 2, .., m\}$ correspond to the set of players and items, respectively. An edge i, j is included if $f_{i,j} > 0$. Then by the same way used by [70], we will prove the following theorem.

Lemma 5.5.5 *There exists a solution $\mathcal{A}'$ to LP5.12, such that $G_{\mathcal{A}}$ is a pseudoforest. (each component of the graph is either a tree or a tree with an extra edge)*

Proof. We have $mn + m + n$ inequalities defining the polytope of feasible solutions of LP5.12. We have mn variables $f_{i,j}$, therefore every solution which is located in the corner of polytope satisfies at least mn inequalities as equalities and there will be at most $m + n$ non-zero variables in these solutions. By the same method used by [70], it is clear to show that if $\mathcal{A}'$ is corresponding solution to a corner of polytope, then $G_{\mathcal{A}'}$ is a pseudoforest. □

We call the solution with the property defined in Lemma 5.5.5 as constrained solution. In [70] it is shown that every constrained solution for LP5.12, can be converted to a solution for IP5.11, such that every agent a_i loses at most one item b_j where $f_{i,j} > 0$.

Lemma 5.5.6 *There exists an allocation in which every agent a_i gets at least* $\mathsf{WMMS}_i - \max_j V_i(\{b_j\})$.

Proof. The polytope of of feasible solutions of LP5.12 is non-empty, because it has at least one solution which is, $f_{i,j} = \dfrac{1}{e_i}$ for every agent a_i and item b_j. Therefore there exists a constrained solution for LP5.12, and using the same method in [70] this solution can be converted to a solution for IP5.11, such that every agent loses at most one item. $\square$

Proofs of Lemmas 5.5.5 and 5.5.6 are omitted and included in the full version.

Now we show that there exists $m'_i = m'_i(c, \epsilon, n, e_1, ..., e_n)$, such that if $m \geq m'_i$, then, $\mathsf{WMMS}_i \geq \dfrac{1}{\epsilon}$ with the probability at least $1 - \dfrac{1}{mn}$. Therefore, whenever $m \geq \max(m'_1, m'_2, ..., m'_n)$, with the probability at least $(1 - \dfrac{1}{mn})^n \geq 1 - \dfrac{n}{nm} = 1 - \dfrac{1}{m}$, for every agent a_i, $\mathsf{WMMS}_i \geq \dfrac{1}{\epsilon}$. According to Lemma 5.5.6, there is an allocation in which every agent gets at least

$$\begin{aligned}\mathsf{WMMS}_i - \max_j V_j(\{b_j\}) &\geq \mathsf{WMMS}_i - 1\\ &\geq \mathsf{WMMS}_i(1 - \frac{1}{\mathsf{WMMS}_i})\\ &\geq \mathsf{WMMS}_i(1 - \epsilon)\end{aligned}$$

Therefore, a $(1 - \epsilon) - \mathsf{WMMS}$ allocation is guaranteed to exist with the probability at least $1 - \dfrac{1}{m}$.

Lemma 5.5.7 *Suppose that for every agent a_j, $\mathbb{E}(\mathcal{D}_j) > c$ for a non-negative constant c. Then for all $0 < \epsilon < 1$, there exists $m'_i = m'_i(c, \epsilon, n, e_1, ..., e_n)$ such that if $m \geq m'_i$, $\mathsf{WMMS}_i \geq \dfrac{1}{\epsilon}$ with the probability at least $1 - \dfrac{1}{mn}$.*

Proof. If there exists a partition of $\mathcal{M}$, $\pi = B_1, B_2, ..., B_n$ in which $V_i(B_j) \geq \dfrac{1}{\epsilon e_i}$ for every agent a_j, then:

$$\mathsf{WMMS}_i \geq e_i \cdot \min(\frac{1}{\epsilon e_i e_1}, ..., \frac{1}{\epsilon e_i e_n}) \geq \min(\frac{1}{\epsilon e_1}, ..., \frac{1}{\epsilon e_n}) \geq \frac{1}{\epsilon} \tag{5.13}$$

Suppose that $m = \frac{\alpha}{\epsilon}$, then for every item b_k and every agent a_j, we assign this item to this agent with the probability e_j. Let $X_{j,k}$ be a random variable that takes the value $V_i(\{b_k\})$ with the probability e_j and 0 otherwise. We have $\mathbb{E}[X_{j,k}] > \frac{c}{e_j}$. Let $X_j = \sum_{k=1}^{m} X_{j,k}$. By setting $nt = \gamma$, we rewrite Equation (5.5) as:

$$\mathbb{P}(|X - \mu| \geq \gamma) \leq 2e^{\frac{-2\gamma^2}{n}} \tag{5.14}$$

$\mathbb{E}(X_j)$ will be at least $\frac{\alpha c}{\epsilon e_j}$. Regarding Equation (5.14), we have:

$$P(|X_j - \mathbb{E}(X_j)| \geq \frac{\alpha c}{\epsilon e_j} - \frac{1}{\epsilon e_i}) \leq 2e^{\frac{-2\epsilon}{\alpha}(\frac{\alpha c}{\epsilon e_j} - \frac{1}{\epsilon e_i})^2}$$

$$\Rightarrow P(|X_j - \mathbb{E}(X_j)| \geq \frac{\alpha c}{\epsilon e_j} - \frac{1}{\epsilon e_i}) \leq 2e^{\frac{-2}{\epsilon\alpha}(\frac{\alpha c}{e_j} - \frac{1}{e_i})^2}$$

Let α_j be the value that $P(|X_j - \mathbb{E}(X_j)| \geq \frac{\alpha_j c}{\epsilon e_j} - \frac{1}{\epsilon e_i}) \leq \frac{1}{mn^2}$, then:

$$
\begin{aligned}
& 2e^{\frac{-2}{\epsilon\alpha_j}(\frac{\alpha_j c}{e_j} - \frac{1}{e_i})^2} \leq \frac{1}{mn^2} \\
& \Rightarrow 2e^{\frac{-2}{\epsilon\alpha_j}(\frac{\alpha_j c}{e_j} - \frac{1}{e_i})^2} \leq \frac{\epsilon}{\alpha_j n^2} \\
& \Rightarrow \frac{2}{\epsilon\alpha_j}(\frac{\alpha_j c}{e_j} - \frac{1}{e_i})^2 \geq ln(\frac{2\alpha_j n^2}{\epsilon}) \\
& \Rightarrow \frac{2\alpha_j c^2}{\epsilon e_j^2} + \frac{2}{\epsilon\alpha_j e_i^2} - \frac{4c}{\epsilon e_i e_j} \geq \ln(\frac{2\alpha_j n^2}{\epsilon}) \\
& \Rightarrow \frac{2\alpha_j c^2}{\epsilon e_j^2} + \frac{2}{\epsilon\alpha_j e_i^2} - ln(\alpha_j) \geq \frac{4c}{\epsilon e_i e_j} + \ln(\frac{2n^2}{\epsilon})
\end{aligned}
$$

Since the right hand side of the inequality is a constant, and the left hand side is an increasing function on its domain, we can find an α'_j such that whenever $\alpha_j \geq \alpha'_j$, this inequality holds. Therefore, $V_i(B_j) \geq \frac{1}{\epsilon e_i}$ with the probability at least $1 - \frac{1}{mn^2}$ whenever $\alpha_j \geq \alpha'_j$. Thus, by choosing proper α such that for all $a_j \in \mathcal{N}$, $\alpha \geq \alpha'_j$, WMMS_i would not be less than $\frac{1}{\epsilon}$ with the probability at least $(1 - \frac{1}{mn^2})^n \geq 1 - \frac{1}{mn}$. $\square$

Chapter 6: Future Work

In this chapter, we will briefly describe the future work which we aim to include in this thesis. As stated earlier in individual chapters, each of these problems have many open directions to be pursued. Here we focus on a subset of these.

- In chapter 3, we provided an approximation algorithms for the EFR and EFX allocation. However, the question of whether an EFX (or even EFR) allocation always exists has remained an important open problem.

- Similar to the fair division of indivisible items, EF1 allocation always exists in the fair division of indivisible chores, and the same question for the EFX allocation is still open. However, in the chore setting, there is no approximation algorithm for the EFX allocation. Giving an approximation algorithm for the EFX allocation of indivisible chores is an interesting direction for the future work.

- As we described in Chapter 4, in a recent breakthrough, after many decades, [9] provide a protocol for the bounded and envy-free cake cutting problem for any number of agents. Our work in [10] gives a protocol for the bounded and envy-free chore division problem for any number of agents. Now, an interesting open problem is giving a bounded and envy-free protocol when we are given a mixture of cake and chore. The simple *cut and choose* protocol solves the problem when the number of agents is two. The problem is even interesting for the case that we have three agents.

- There is a large gap between the bound of the protocol proposed by [9] for the cake cutting problem and its lowerbound [100]. Improving any of these bounds is an interesting direction for the future work.

- An important variant of the fair allocation problem of the indivisible goods is when the goal is maximizing the least value that any agent obtains from the allocation. Asadpour and Saberi [13] give the first polynomial time algorithm for this problem that approximates the optimal solution within a factor of $O(\sqrt{n}\log^3 n)$ in the additive setting. This was later improved by Chakrabarty, Chuzhoy,and Khanna [101] to an $O(m^\epsilon)$ approximation factor. A very interesting open problem is giving a constant approximation algorithm for this problem.

- A special case of the above problem in which the valuation of every agent for an item is either 0, or a fixed value is called *the Santa Claus problem*. This problem was first introduced by Bansal and Sviridenko [102]. In this paper, they give an $O(\log\log n/\log\log\log n)$ approximation algorithm for this problem that runs in polynomial time. Later Feige [103] showed that the objective value of the problem can be approximated within a constant factor in polynomial time. This was later turned into a constructive proof by Annamalai *et al.* [104]. Improving their approximation ratio is an interesting future work.

- For allocating indivisible goods, *spliddit*, a popular fair division website, uses the maximum Nash welfare allocation (the allocation that maximizes the product of utilities). [105] provide the first constant approximation algorithm for maximizing the geometric mean of the agents' valuation. Improving and generalizing their work is an interesting direction for the future work.

- In Chapter 2, we provided a better guarantee for the approximate maximin share. As we described, the works by [20] and [19] have considered the minimax share guarantees. Further exploring the minimax share guarantees is a direction for the future research.

Bibliography

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

[2] David Kurokawa, Ariel D Procaccia, and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM (JACM)*, 65(2):8, 2018.

[3] Mohammad Ghodsi, MohammadTaghi HajiAghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. Fair allocation of indivisible goods: Improvement and generalization. EC 2018.

[4] Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 125–131, 2004.

[5] Georgios Amanatidis, Apostolos Ntokos, and Evangelos Markakis. Multiple birds with one stone: Beating $1/2$ for efx and gmms via envy cycle elimination. In *arXiv preprint arXiv:1909.07650*, 2019.

[6] Martin Gardner. *Aha! Aha! insight*, volume 1. Scientific American, 1978.

[7] S Garfunkel. For all practical purposes social choice. *COMAP*, 1988.

[8] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for four agents. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 454–464. ACM, 2016.

[9] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 416–427. IEEE, 2016.

[10] Sina Dehghani, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Hadi Yami. Envy-free chore division for an arbitrary number of agents. In *SODA*, 2018.

[11] Alireza Farhadi, MohammadTaghi Hajiaghayi, Mohammad Ghodsi, Sebastien Lahaie, David Pennock, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. Fair allocation of indivisible goods to asymmetric agents. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1535–1537. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[12] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms (TALG)*, 13(4):52, 2017.

[13] Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 114–121. ACM, 2007.

[14] Sylvain Bouveret and Michel Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems*, 30(2):259–290, 2016.

[15] Lester E Dubins and Edwin H Spanier. How to cut a cake fairly. *American mathematical monthly*, pages 1–17, 1961.

[16] Hugo Steinhaus. The problem of fair division. *Econometrica*, 16:101–104, 1948.

[17] Duncan K Foley. Resource allocation and the public sector. *YALE ECON ESSAYS, VOL 7, NO 1, PP 45-98, SPRING 1967. 7 FIG, 13 REF.*, 1967.

[18] Steven J Brams and Alan D Taylor. An envy-free cake division protocol. *American Mathematical Monthly*, pages 9–18, 1995.

[19] Siddharth Barman and Sanath Kumar Krishna Murthy. Approximation algorithms for maximin fair division. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 647–664. ACM, 2017.

[20] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. Algorithms for max-min share fair allocation of indivisible chores. In *AAAI*, volume 17, pages 335–341, 2017.

[21] Georgios Amanatidis, Georgios Birmpas, George Christodoulou, and Evangelos Markakis. Truthful allocation mechanisms without payments: Characterization and implications on fairness. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 545–562. ACM, 2017.

[22] Georgios Amanatidis, Georgios Birmpas, and Evangelos Markakis. On truthful mechanisms for maximin share allocations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 31–37. AAAI Press, 2016.

[23] David Kurokawa, Ariel D Procaccia, and Junxing Wang. When can the maximin share guarantee be guaranteed? In *AAAI*, volume 16, pages 523–529, 2016.

[24] R Paes Leme. Gross substitutability: An algorithmic survey. *preprint*, 2014.

[25] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 610–618. ACM, 2005.

[26] Uriel Feige. On maximizing welfare when utility functions are subadditive. *SIAM Journal on Computing*, 39(1):122–142, 2009.

[27] Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 461–471. IEEE, 2007.

[28] Uriel Feige and Jan Vondrak. Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 667–676. IEEE, 2006.

[29] Michal Feldman, Nick Gravin, and Brendan Lucier. Combinatorial auctions via posted prices. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 123–135. Society for Industrial and Applied Mathematics, 2015.

[30] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74. ACM, 2008.

[31] Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 125–131. ACM, 2004.

[32] MohammadHossein Bateni, Mohammadtaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Transactions on Algorithms (TALG)*, 9(4):32, 2013.

[33] Daniel Golovin. Max-min fair allocation of indivisible goods. 2005.

[34] Leah Epstein and Asaf Levin. An efficient polynomial time approximation scheme for load balancing on uniformly related machines. *Mathematical Programming*, 147(1-2):1–23, 2014.

[35] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[36] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 649–658. IEEE, 2012.

[37] Niv Buchbinder, Moran Feldman, Joseph Seffi, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization.

[38] Satoru Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

[39] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics*, pages 246–257. Springer, 2010.

[40] Gunhee Kim, Eric P Xing, Li Fei-Fei, and Takeo Kanade. Distributed cosegmentation via submodular optimization on anisotropic diffusion. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 169–176. IEEE, 2011.

[41] Andreas Krause. Sfo: A toolbox for submodular function optimization. *The Journal of Machine Learning Research*, 11:1141–1144, 2010.

[42] Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 323–332. ACM, 2009.

[43] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pages 234–243. Springer, 1978.

[44] George Christodoulou, Annamária Kovács, and Michael Schapira. Bayesian combinatorial auctions. In *International Colloquium on Automata, Languages, and Programming*, pages 820–832. Springer, 2008.

[45] Kshipra Bhawalkar and Tim Roughgarden. Welfare guarantees for combinatorial auctions with item bidding. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 700–709. Society for Industrial and Applied Mathematics, 2011.

[46] Liad Blumrosen and Shahar Dobzinski. Welfare maximization in congestion games. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 52–61. ACM, 2006.

[47] Vasilis Syrgkanis. Bayesian games and the smoothness framework. *arXiv preprint arXiv:1203.5155*, 2012.

[48] Michal Feldman, Hu Fu, Nick Gravin, and Brendan Lucier. Simultaneous auctions are (almost) efficient. In *STOC 2013*.

[49] Hu Fu, Robert Kleinberg, and Ron Lavi. Conditional equilibrium outcomes via ascending price processes with applications to combinatorial auctions with item bidding. In *ACM EC 2012*.

[50] Igal Milchtaich. Congestion games with player-specific payoff functions. *Games and economic behavior*, 13(1):111–124, 1996.

[51] Achim Bachem and Walter Kern. *Linear programming duality*. Springer, 1992.

[52] Alireza Farhadi, Mohammad Ghodsi, Mohammad Taghi Hajiaghayi, Sebastien Lahaie, David Pennock, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. Fair allocation of indivisible goods to asymmetric agents. *Journal of Artificial Intelligence Research*, 64:1–20, 2019.

[53] Masoud Seddighin, Hamed Saleh, and Mohammad Ghodsi. Externalities and fairness. In *Proceedings of the 2019 World Wide Web Conference*, pages 538–548, 2019.

[54] Mohammad Ghodsi, MohammadTaghi HajiAghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. Fair allocation of indivisible goods: Improvements and generalizations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 539–556, 2018.

[55] John P Dickerson, Jonathan R Goldman, Jeremy Karp, Ariel D Procaccia, and Tuomas Sandholm. The computational rise and fall of fairness. In *AAAI*, pages 1405–1411. Citeseer, 2014.

[56] Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi, and Toby Walsh. Fair allocation of indivisible goods and chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 53–59, 2019.

[57] David Kurokawa, Ariel D Procaccia, and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM*, 65(2):8, 2018.

[58] Benjamin Plaut and Tim Roughgarde. Almost envy-freeness with general valuations. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2584–2603. SIAM, 2018.

[59] John F Nash Jr. The bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 155–162, 1950.

[60] Bhaskar Ray Chaudhury, Tellikepalli Kavitha, Kurt Mehlhorn, and Alkmini Sgouritsa. A little charity guarantees almost envy-freeness. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2658–2672, 2020.

[61] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation*, 7(3):12, 2019.

[62] Hugo Steinhaus. The problem of fair division. *Econometrica*, 16(1), 1948.

[63] Hal R Varian. Equity, envy, and efficiency. 1973.

[64] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *Proceedings of the 57th Annual Symposium on Foundations of Computer Science*, pages 416–427. IEEE, 2016.

[65] Sina Dehghani, Alireza Farhadi, MohammadTaghi HajiAghayi, and Hadi Yami. Envy-free chore division for an arbitrary number of agents. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2564–2583, 2018.

[66] Ioannis Caragiannis, Nick Gravin, and Xin Huang. Envy-freeness up to any item with high nash welfare: The virtue of donating items. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 527–545, 2019.

[67] Georgios Amanatidis, Georgios Birmpas, and Evangelos Markakis. Comparing approximate relaxations of envy-freeness. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 42–48, 2018.

[68] Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

[69] Jack Robertson and William Webb. Cake-cutting algorithms: Be fair if you can. 1998.

[70] Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *ACM SIGecom Exchanges*, 5(3):11–18, 2005.

[71] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding fair and efficient allocations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 557–574, 2018.

[72] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms (TALG)*, 13(4):1–28, 2017.

[73] Jugal Garg, Peter McGlaughlin, and Setareh Taki. Approximating maximin share allocations. In *Proceedings of the 2nd Symposium on Simplicity in Algorithms*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[74] Elisha Peterson and Francis Edward Su. N-person envy-free chore division. 2009.

[75] Steven Brams, Alan Taylor, and William Zwicker. A moving-knife solution to the four-person envy-free cake-division problem. *Proceedings of the American Mathematical Society*, 125(2):547–554, 1997.

[76] Amin Saberi and Ying Wang. Cutting a cake for five people. *AAIM*, 9:292–300, 2009.

[77] Elisha Peterson and Francis Edward Su. Four-person envy-free chore division. *Mathematics Magazine*, 75(2):117–122, 2002.

[78] Egor Ianovski. Cake cutting mechanisms. *arXiv preprint arXiv:1203.0100*, 2012.

[79] Julius B Barbanel and Alan D Taylor. Preference relations and measures in the context of fair division. *Proceedings of the American Mathematical Society*, 123(7):2061–2070, 1995.

[80] Costas Busch, Mukkai S Krishnamoorthy, and Malik Magdon-Ismail. Hardness results for cake cutting. *Bulletin of the EATCS*, 86:85–106, 2005.

[81] Jeff Edmonds and Kirk Pruhs. Cake cutting really is not a piece of cake. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 271–278. Society for Industrial and Applied Mathematics, 2006.

[82] Erel Segal-Halevi, Avinatan Hassidim, and Yonatan Aumann. Waste makes haste: Bounded time protocols for envy-free cake cutting with free disposal. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 901–908. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[83] David Kurokawa, John K Lai, and Ariel D Procaccia. How to cut a cake before the party ends. In *AAAI*, 2013.

[84] Claudia Lindner and Jörg Rothe. Cake-cutting: Fair division of divisible goods. In *Economics and Computation*, pages 395–491. Springer, 2016.

[85] Ariel D Procaccia. Cake cutting: not just child's play. *Communications of the ACM*, 56(7):78–87, 2013.

[86] Ariel D Procaccia. Cake cutting algorithms. In *Handbook of Computational Social Choice, chapter 13*. Citeseer, 2015.

[87] Haris Aziz. Computational social choice: Some current and new directions. In *IJCAI*, pages 4054–4057, 2016.

[88] Oleg Pikhurko. On envy-free cake division. *The American Mathematical Monthly*, 107(8):736–738, 2000.

[89] Yiling Chen, John K Lai, David C Parkes, and Ariel D Procaccia. Truth, justice, and cake cutting. *Games and Economic Behavior*, 77(1):284–297, 2013.

[90] Yuga Julian Cohler, John Kwang Lai, David C Parkes, and Ariel Procaccia. Optimal envy-free cake cutting. 2011.

[91] Gerhard J Woeginger and Jiří Sgall. On the complexity of cake cutting. *Discrete Optimization*, 4(2):213–220, 2007.

[92] Ioannis Caragiannis, John K Lai, and Ariel D Procaccia. Towards more expressive cake cutting. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 127, 2011.

[93] Reza Alijani, Majid Farhadi, Mohammad Ghodsi, Masoud Seddighin, and Ahmad S. Tajik. Envy-free mechanisms with minimum number of cuts. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 312–318, 2017.

[94] Steven J. Brams, D. Marc Kilgour, and Christian Klamler. Maximin envy-free division of indivisible items. In *Group Decis Negot*, volume 4, pages 115–131, 2017.

[95] Moshe Babaioff, Noam Nisan, and Inbal Talgam-Cohen. Competitive equilibria with indivisible goods and generic budgets. In *arXiv preprint arXiv:1703.08150*, 2017.

[96] Steven J Brams, Michael A Jones, and Christian Klamler. Proportional pie-cutting. *International Journal of Game Theory*, 36(3-4):353–367, 2008.

[97] Jack M Robertson and William A Webb. Extensions of cut-and-choose fair division. *Elemente der Mathematik*, 52(1):23–30, 1997.

[98] Kevin McAvaney, Jack Robertson, and William Webb. Ramsey partitions of integers and pair divisions. *Combinatorica*, 12(2):193–201, 1992.

[99] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[100] Ariel D Procaccia. Thou shalt covet thy neighbor's cake. In *In Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 239–244, 2009.

[101] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *FOCS 2009*.

[102] Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In *STOC 2006*.

[103] Uriel Feige. On allocations that maximize fairness. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 287–293. Society for Industrial and Applied Mathematics, 2008.

[104] Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial algorithm for restricted max-min fair allocation. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1357–1372. SIAM, 2015.

[105] Richard Cole and Vasilis Gkatzelis. Approximating the nash social welfare with indivisible items. In *STOC*, pages 371—-380, 2015.