FAIR DIVISION OF INDIVISIBLES THROUGH THE COMPUTATIONAL LENS

BY

RUCHA KULKARNI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Doctoral Committee:

Assistant Professor Ruta Mehta, Chair
Professor Chandra Chekuri
Professor Jeff Erikson
Professor Ariel Procaccia, Harvard

To my mother and father, as a small token of my eternal gratitude to them.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Aristotle's maxim 'Equals should be treated equally, and unequals unequally, in proportion to relevant similarities and differences' was the philosophical blueprint concretized by economists and mathematicians into the theory of fair division. Thorough explorations for almost a century starting from the 1940s resulted in the concrete fair division problem and its extensive analysis. This work resulted in numerous definitions and solution concepts that were meaningful in capturing the seemingly subjective notion of fairness. Notably, the focus of most of this work was not on the computational requirements of any methods that achieved the fair solutions, but was on the properties of the final solution.

During the same time in the 1940s, mathematicians Steinhaus, Banach and Knaster were starting to focus on this very ignored challenge of fair division, albeit in a different manner than the way computer science studies it. They initiated the study of constructive methods to prove the existence of fair allocations of a single resource, termed a *cake*. Their inspiration for this dates back longer than Aristotle, to the Bible. Among several references to fair allocation problems in the Bible is Solomon's cut-and-choose protocol for a proportional division among two agents. Steinhaus posed the problem of extending this to an arbitrary number of agents, to Banach and Knaster. Their positive answer was the last diminisher method, which set in motion the quest for such protocols for fairly dividing a cake, and a further computational study of these, for example, bounding the number of cuts on the cake to obtain the fair solution.

Simultaneously, computer science was also starting to focus on computational investigations. The visionary John von Neumann, in his 1952 paper, introduced the notion of a polynomial time algorithm as a desirable characteristic. Subsequently, the rapid growth in computing technology made efficient, that is, polynomial time, computation a fundamental problem in all areas of computer science. The design and analysis of algorithms, and complexity theory - a systematic language for analyzing the limits of computability, emerged as separate areas, and became central to computer science. Further, paradigms like approximation algorithms were founded out of the view that efficient computation was a vital requirement, even if the final solution was less optimal.

The advent of the Internet and the industrial revolution changed the nature of human interaction, and the nature of real world problems. Today, in the 21st century, the pressing scientific challenges are largely fair allocation problems, arising due to (i) limited natural resources necessary for survival, (ii) and man-made problems contributing to this scarcity, necessitating fair allocations of the resources as well as the responsibilities to curb the man-made reasons. Algorithms have become the heart of how the economy functions, humans interact, and therefore how solutions are deployed. The computability of fair division was naturally drawn into spotlight for both economists and computer scientists, and is also the broad subject of this thesis.

The computational study of fair division requires specifying several aspects of the problem formally. In Section 1.1, we will see how fair division evolved into a rich mathematical theory, and how fairness came to have multiple formal definitions. This thesis is divided into three parts, out of which the first is based on the joint work [1], and explores the Nash social welfare (NSW) notion. NSW was independently discovered as a meaningful objective by three different communities - as a solution of the bargaining problem in classic game theory, as a solution for proportional fairness in networking, and as an equilibrium concept in economics, termed as the competitive equilibrium with equal incomes (CEEI). Nash welfare is well-studied by the fair division community as well. We extend the understanding of this objective to more general settings, namely when the individuals to allocate resources to are 'unequal', a setting termed the asymmetric agent entitlements case, and when they have more complex utility functions, called the submodular functions. We describe the setting and our results in Chapter 3.

The second part of this thesis studies another popular notion of fairness, the Maximin share (MMS). This is based on the joint works [2] and [3]. Maximin share was introduced by [4] as a relaxation of two other fairness notions, proportionality and envy-freeness. These notions were defined and studied for a setting where the resources were assumed to be divisible, meaning they could be fractionally allocated, for example, a cake. The Maximin share was particularly defined for the setting when the resources are indivisible, for example automobiles, university seats, or computing resource slots on networks. We initiate the study of the computability of allocations that achieve the MMS guarantees, for more general settings than previously studied. Specifically, we study the case when some of the resources could be chores, meaning negatively valued by the individuals. Further, each resource could be a good to some and simultaneously a chore for others. We also add the requirement that the solutions guarantee Pareto optimality (PO), meaning it should not be possible to improve the utility of one individual without having to decrease some other individual's. Details and a formal discussion are in Chapter 4. We also initiate the study of fair allocations under the notion of MMS, for the setting where the utility functions of the individuals are something called the assignment, or OXS valuations. A definition of and the motivation to study this setting, along with a formal discussion of our results, are in Chapter 5.

The final part of the thesis has a different theme, arising from our belief that fairness notions will be useful in allocation problems beyond the typical settings where they are popularly used. Towards establishing this formally, we first look at the world of routing problems, popularly studied as congestion games. Here players wish to travel between two nodes on a given map, and their cost to travel is a function of the roads they take, and the congestion, or the number of players, on each road. Allocating routes is a natural fairness problem, and we apply definitions related to envy-freeness, a notion primarily defined for allocating non-shareable resources, to this setting. A formal discussion and details are in Chapters 6 and 7. This discussion is based on joint works [5]

and [6].

Organization. The rest of the thesis is organized as follows. In the remainder of this chapter, we first overview how fair division evolved from a philosophical maxim into a rich mathematical theory (Section 1.1). Here we introduce the notation and formal definitions of concepts that will be useful in the technical discussion. We then briefly overview routing games in Section 1.2, the popular notions studied in their classical theory, and motivate fairness here. In the next chapter 2, we review the state of the art literature, the important and breakthrough works that led to these, and summarize our results, separately for each part of the thesis.

The technical discussion of our work follows in the next chapters. Chapter 3 is a detailed discussion of our work on the Nash welfare problem. Chapters 4 and 5 is a discussion of the second part of the thesis, the Maximin share fair division problem. The final part, on fairness applied to routing settings, is detailed in Chapters 6 and 7. The organization of each of these chapters can be found within the corresponding chapter. These chapters are independent of each other, and can be read in any order. We conclude with Chapter 8, overviewing some future directions that can be pursued, to take our work ahead.

1.1 THE EVOLUTION OF FAIR DIVISION

Fair division is a complex theory deliberated with deep thoughtful consideration. There are several excellent works that each explain how different parts of this theory came into being and the current state of the art literature. For example, see [7] for the economic considerations developing various versions of the fair division problem, [8, 9, 10] for cake-cutting - the popular term for the fair division problem on non-shareable resources that can be fractionally allotted, and [11, 12] for the distribution of non-shareable resources that need to be integrally allocated. This section only surveys some of the basic reasoning through toy examples and questions, attempting to give a glimpse of the complexities of this modeling process. At the end of this section is a discussion of the broad directions of research undertaken by computer scientists working in this area.

There are several aspects of Aristotle's celebrated maxim that need to be made precise to transform it into a mathematical theory. To begin with, one needs to formalize how 'equality' of individuals should be measured. What are the 'relevant similarities and differences' of the individuals, and how should they be quantified? We describe a few real world settings to show that there can be multiple ways to do this.

Voting. 'One man one vote' is the foundational principle of democracy. This rule treats every adult as equal, irrespective of any individual characteristic, and has very few exceptions (for exam-

ple, insanity arguments or criminal records). However, one could argue that there could be reasons why one individual's vote should matter more than another. For example, one individual could be making an informed choice, and another may be voting randomly. One individual could be living in a different region and does not plan to move back, while another has been and will be living in the region whose representative is being decided through the vote. One individual may have 'earned' the right to a higher valued vote, by being a 'more' contributing member of the society - for instance, the votes of teachers, army veterans and doctors 'deserve' to be valued higher than those of billionaires.

Analogous arguments can be made about any other rights guaranteed to every individual, for example the right to education.

Combating climate change. Climate change is perhaps the most severe challenge today. Taking measures to combat it has become a necessity. The Montreal Protocol was the first international agreement that implicitly tackled climate change, by discussing the elimination of chlorofluorocarbons (CFCs), as they harm the ozone layer. The United Nations Framework Convention on Climate Change (UNFCCC) was a landmark forum that explicitly addressed climate change. Participating nations of the UNFCCC meet annually to discuss measures to stabilize greenhouse gas concentration in the atmosphere. These annual meetings essentially discuss a fair treatment problem - what percentage reduction in greenhouse gas emissions should each nation pledge to undertake that year? It can be argued that this decision be based on the country's contribution to the problem, thus higher emitters of these gases should take higher responsibilities. Or one can measure the needs of countries, for example, developing countries should not be asked to reduce their emissions. Or one can ask every country, irrespective of any characteristic, to reduce their emissions by the same percentage amount.

Splitting assets. Consider the frequently occurring situation where the assets of an estate must be distributed among the heirs after the current owner has deceased. Should one consider the needs of each heir, by measuring their independent wealth, and give larger shares to the poorest one? Or should we do the opposite, meaning reward the richer heir, as they indicate the ability to handle the assets in a better way? Or should we reward the heir who diligently took care of the current owner in their final years?

And finally, a simpler toy example where there is a single natural direction.

Cookies and Friends. Suppose there are two friends, Alice and Bob who wish to share two cookies, one macademia white chocolate flavored, and the other chocolate chip. Alice is allergic to all nuts, hence will throw away any amount of the macademia cookie she gets. Bob is not allergic,

but prefers eating the chocolate chip one as well. What percentage of each cookie should each get to call the distribution an equal treatment? To solve this question, the first step is to decide what aspects of Alice and Bob must be taken into consideration. Should Alice's allergy be a part of the measure that will decide the equality of Bob and Alice? Should Bob's preferences be considered? If both are considered, do they matter equally?

The various alternative ways proposed in these examples to measure the equality of individuals each have their own merit, hence must be chosen as the 'right' measure based on the application. For modeling these in a single mathematical problem, the following simple notion of a *utility function* is defined for each individual. Individuals are treated equally or unequally based only on these, and all other characteristics are deemed irrelevant for the fair allocation decision. Particularly, individuals with identical utility functions must be allocated identical resources.

Utility functions. Every individual is associated with something called a utility or valuation function. There are two ways in which this notion can be defined, resulting in two types of functions. The first is called the *cardinal utility functions*. Such a function assigns a real value to every possible allocation that the individual can receive. The second type of functions are called the *ordinal utility functions*. These are comparison operators that given a set of possible allocations, can rank them from the most preferred to the least.

Individuals are assumed to be rational, and prefer receiving allocations with higher cardinal utility function values, or the most preferred allocation according to the ordinal function.

The setting of the fair division problem will be key in deciding whether to model it as one with a cardinal or an ordinal functions. Every cardinal function implies an ordinal function, but these functions are generally harder to elicit from the individuals than ordinal functions, and require more resources to store and process in the fair division algorithms. Therefore it is beneficial to not use them when not essential. For instance, in the Cookies and Friends example, The ordinal utility function of both Alice and Bob would be identical. However, to treat them fairly, one would want to consider both Alice and Bob's requirements, but would want Alice's allergy to impact the fairness decision more than Bob's dislike. Therefore using cardinal functions is meaningful here. But in instances when fairness is measured using envy, meaning individuals feel fairly treated if they each prefer their own share compared to those of the other individuals, then an ordinal function is sufficient.

We will go over the computational considerations in modeling these functions, for example their size, analytical properties that make them amenable to polynomial time computation, and how they are specified to the fairness algorithms, after discussing how all the other parameters in the fair division problem were modeled.

Resources. The second task in this pursuit of modeling Aristotle's maxim is creating mathematical handles for the resources that are to be allocated. In the voting example, the number of votes that decide the political representative can be the resources, with the fairness problem being how many must be assigned to each individual. In the climate change example, the central problem, the total amount of greenhouse gas emissions to be reduced in a year are the 'resources', the problem is to decide how much of these will be assigned to each country. The countries will have negative valued utility functions, with worse values for higher amounts assigned to them. In the last two examples, the resources are clear - assets of the estate, and the two cookies.

These examples only capture settings where the resources are *non-shareable*, meaning the allocations to distinct individuals need to be disjoint. Fair division theory goes beyond such settings, for example to capture fairness problems of shareable resources, for instance public infrastructure like roads, parks, etc., or dynamic resources with utilities that change over time, like lab slots for running experiments. The discussion of these is beyond the scope of this article.

For mathematical reasons, non-shareable resources are classified into two types - divisible and indivisible. The former are those that can be fractionally divided and allocated to multiple individuals, and the latter are those that must be integrally allocated. To see a small instance of why these types must be modeled separately, we point to the Cookies and Friends example again. If the cookies were constrained to be integrally allocated, then this problem becomes very different, with a natural solution - if the fairness notion requires both Alice and Bob to have positive utilities, then Alice must be given the chocolate chip cookie. On the other hand, integral allocation constraints, when the number of items are large, can be far harder to solve than the fractional assignment problems in the same setting. As empirical evidence, we point to the computational hardness of integer linear programming problems as opposed to their fractional relaxations.

Divisible resources are denoted by real intervals. A large part of the literature on fair division of these assumes a single divisible resource, which without loss of generality is denoted by the real interval $[0, 1]$. Individuals can be allotted disjoint sub-intervals, and each agent can receive multiple sub-intervals. Consequently their utility functions specify their preferences for every sub-interval. Popularly, the resource is called a *cake*, and the sub-intervals are called *pieces*. This fair division problem is called the cake-cutting problem, which informally is as follows.

Cake-cutting problem: Given a cake, and n agents with their utility functions over the cake, how can we fairly divide the cake among the agents?

Indivisible resources are denoted by a set. Individuals can be allotted disjoint subsets of this set, hence their utility functions specify preferences for every subset of the set of resources. The fair division problem with indivisible resources is as follows.

Fair division with indivisible resources: Given n agents, m resources, and agent utility functions over the resources, how can we fairly partition the resources among the agents?

Indivisible resource settings can be further classified, based on whether the set of resources are all *goods*, meaning non-negatively valued by all the individuals, or are all *chores*, meaning each is non-positively valued by all, or are *mixed*, meaning every resource can be non-negatively valued by some and negatively by others.

This thesis focuses mostly on indivisible resources. We will limit the discussion of notions in the divisible setting to the topics that influence the theory of indivisible resources.

Fairness definitions

The informal fair division problem descriptions point us to our next task in the quest of formalizing fair division - qualifying what is fair treatment. More specifically, what is a mathematical definition of fairness? Initial research on the cake-cutting problem led to the following two fundamental notions.

Proportionality. Solomon's solution in the Bible for dividing a single object fairly into two disputing individuals or agents, was the cut-and-choose protocol - one agent cuts the object into two as per their own utility function, and the other agent chooses their favorite part first. The argument for fairness is, the agent who cuts will make as equal pieces of the object as possible, thus receiving a share they value half of their value for the whole object. The other agent will value their chosen piece more than the one they rejected, hence also receives at least half of their value for the object.

The implicit notion of fairness being used here is proportionality. Indeed, Steinhaus formally proposed proportionality as the notion for fairness, in his introductory paper initiating the cake-cutting problem [13]. Proportionality is defined as follows.

Definition 1.1.1 (Proportionality [13])**.** *Given a cake and n agents with utility functions for the cake, a division of the cake among the agents is termed proportional, if every agent values their share at least $1/n$ times their value for the entire cake.*

Envy-freeness [14]. Suppose a division of a cake satisfies the property that every individual prefers their own share compared to anyone else. Individuals are said to have *no envy* for any other individual in such a solution, and such a division of a cake is said to satisfy the property termed envy-freeness. Gamow and Stern [14] defined fairness as envy-freeness, that is, a fair division of a cake is equivalent to an envy-free division.

Envy-freeness is a stronger property than proportionality, as the former implies the latter. The converse may not be true. For instance, the result of the cut-and-choose protocol is proportional, but may have the individual who cuts envying the individual who chooses.

Competitive equilibrium with equal incomes (CEEI). A drawback of both envy-freeness and proportionality is that they do not guarantee an allocation of the resources to agents who value them the most, in the utility maximization sense, as illustrated below.

For example, consider the Cookies and Friends example from the beginning of this section. Suppose the utility function was instead as follows. Alice has utility 10 for the chocolate chip cookie, and 0 for the white chocolate macademia. Bob has value 5 for each. Now consider the following two alternative allocations. The first one gives both half of each cookie. Irrespective of the utilities, this is an identical distribution, hence both proportional and envy-free. The utilities of the two friends in this division are 5 each, with a total utility of 10. Consider the alternate division that gives the chocolate chip cookie to Alice, and the other one to Bob. Each receives the cookie they prefer over the other, hence the division is both envy-free and proportional. This also yields utilities of 10 and 5 for their shares, and a total utility of 15.

The question then is, can we obtain a division of the resources that is envy-free, hence proportional, and also ensures the sum of the individual utilities is maximized?

The answer to this question is negative, that is, there may be instances where the set of envy-free divisions and utility maximizing divisions do not intersect. A simple counterexample is the above instance where Bob's utilities are now modified to 100 for both the cookies.

There is another notion for *optimizing* the allocation of resources (and not maximizing the sum of utilities), known as Pareto optimality. This is when no other division can ensure a higher utility share to some individual without decreasing the utility of any other individual.

Definition 1.1.2 (Pareto optimality)**.** *An allocation is called Pareto optimal if there is no other allocation where at least one agent receives a strictly higher valued share, and every other agent receives at least an equal valued share.*

Any allocation that satisfies either of these properties - maximum sum of utilities (MSW), or Pareto optimality, is called *economically efficient.* Note that the former implies the latter. The key difference between the two properties is *scale-freeness.* MSW incentivizes individuals to magnify their utility functions arbitrarily. As individuals are assumed to be rational and desire to maximize their own utility, with no consideration to that of others, such concepts suffer the serious drawback that if applied in real life interactions, they do not ensure truthful reporting of utility functions.

From the perspective of fair division, the question then is, is there an allocation that satisfies envy-freeness and PO?

A positive answer to this is the compelling notion of competitive equilibrium with equal incomes (CEEI). Defined and shown to exist by [15], the idea is that every individual is given an equal amount of an artificial currency to spend on the resources. The solution, or the equilibrium, is a price vector, specifying the price for every possible share of the resource, and a division of the

entire cake such that every individual spends all their money buying the most valued share they could.

As individuals were given the same amount of money, the shares bought by every individual are affordable to every other individual. And as individuals have bought shares that have the highest utility among all their affordable shares, they do not envy any other individual's shares.

The solution also satisfies Pareto optimality. One can show this by arguing that any solution that contradicted this property of the CEEI solution would instead have been the CEEI solution.

These three notions form the three foundational pillars of the cake-cutting problem. The indivisible resource problem, however, requires renewed explorations. To see why, consider the simple example where one indivisible resource must be allocated among two agents. Neither a proportional nor an envy-free, hence nor a CEEI solution exists here. Researching new fairness notions useful for this setting was a new line of research, resulting in the following definitions.

Maximin share (MMS) **[4].** The cut-and-choose protocol, the key motivation behind introducing proportionality, is fair because it ensures the following - if an individual themselves make a division of the resources and chooses last, then they are only guaranteed the share with the lowest utility, hence cut in a way that maximizes the value of this worst share.

Maximin share is the value assured if the cut-and-choose experiment were simulated in the indivisible resource setting. Formally, the MMS value of an agent is defined as follows.

Definition 1.1.3 (Maximin share (MMS) [4].)**.** *Given a set of m indivisible resources, n agents, and one agent's utility function $v(.)$ specifying their utility for every subset of the resources, this agent's* MMS *value is the maximum possible utility of the minimum valued bundle, maximized over all possible divisions of the resources into n bundles.*

Analogous to the cut-and-choose protocol's guarantee, a division of resources among a set of agents is said to satisfy maximin fairness, if every agent receives a share of value at least equal to their maximin share value.

Interestingly, maximin share allocations may not exist for every fair division instance, as shown by [16]. Therefore, the definition was relaxed to find allocations that ensure everyone α times their MMS value, for the best constant $\alpha < 1$ for which such allocations do exist. There are two directions in which this relaxed problem can be pursued. The first is an existence problem - find the largest constant α such that α-maximin fair allocations exist for all fair division problem instances. The second is a computational problem - design a polynomial time algorithm, that computes for every instance, an α-maximin fair allocation for the best α for the instance, meaning an $(\alpha+\epsilon)$-fair allocation does not exist for this instance, for any $\epsilon > 0$.

α-maximin fairness is a scale-free property. This means that if the agent utility functions are scaled by any constant factor, an α-maximin fair allocation for the original problem is also α-maximin fair in the scaled setting.

Relaxations of envy-freeness. As envy-free divisions of indivisible resources may not exist, extending the notion to this setting involved relaxing the definition. The challenge is then to define and obtain approximate solutions by relaxing the envy definition as slightly as possible. The first work to start this study was [17]. Their focus was designing approximation algorithms that satisfied envy-freeness as closely as possible, rather than defining new relaxed notions. One of their algorithms that they termed *bounded the maximum envy,* was explicitly termed and proposed as the fairness notion EF1, by Budish [4]. This latter work proposed two key relaxations of envy-freeness. These definitions have the desirable property that they can be verified, hence could also be computed, using ordinal utility functions.

Definition 1.1.4 (Envy-free up to one item (EF1) [4].)**.** *An allocation of resources is said to be envy-free up to one item, if for every individual a, there is some item that can be taken away from any other individual b, after which a values their own share more than they value the share of b.*

Definition 1.1.5 (Envy-free up to any item (EFx) [4].)**.** *An allocation of resources is said to be envy-free up to any item, if for every individual a, upon taking away any item from any other individual b, a values their own share more than they value the share of b.*

The algorithm proposed by [17] shows that EF1 allocations always exist, even in very general settings, for example when one does not assume any restrictions on the agent utility functions, other than monotonically non-decreasing utilities over increasing sets of the resources. On the other hand, the problem of deciding whether EFx allocations exist, even in special cases, such as a setting with 4 agents with linear utility functions (utility for a set of resources equals the sum of utilities for the individual resources in the set) is an important open problem.

Nash welfare. Proposed as a solution to the bargaining problem [18], the Nash welfare is a value associated with every possible allocation of the resources, defined as follows.

Definition 1.1.6 (Nash welfare (NSW) [18].)**.** *The Nash welfare of an allocation is the geometric mean of all the agent's utility values for their shares in the allocation.*

A solution that maximizes the Nash welfare value over all possible allocations is said to be Nash optimal. [19] showed several properties of a Nash optimal allocation, giving compelling reasons why it should be accepted as a solution to the fair division problem. For instance, these allocations satisfy EF1, and are Pareto optimal. Further, if the indivisible resources are relaxed and allowed

to be fractionally allocated, the Nash optimal solution coincides with the CEEI solution for this divisible setting.

Computationally, computing Nash optimal solutions is NP-hard. Therefore, the key open problem is finding α-approximately Nash optimal allocations, or those whose Nash welfare value multiplied by some constant α equals the highest Nash welfare value, for as small an $\alpha > 1$ as possible.

α-Nash optimality satisfies scale-freeness of the utility functions. That is, if agent utility functions are scaled by any distinct constant factors, an α-Nash optimal allocation for the original problem is also α-Nash optimal in the scaled setting. Particularly, the set of Nash optimal solutions remains the same.

We describe one last parameter in the formalization of the fair division problem, defined due to certain drawbacks that arise from the way that the other parameters were formalized.

Agent weights. Recall the part of Aristotle's maxim of treating unequals unequally. One way to capture agent inequalities, as argued in the motivation for their definition, is to have the agents' utility functions reflect the inequality. However, if the underlying ordinal utility functions of the agents are identical, and the fairness notion being considered depends only on these, or is scale-free, then the utility functions will be insufficient. To resolve this, we associate a new parameter, termed the *agent weight or entitlement*. This is a scalar value, and the idea is to divide the resources fairly according to these weights, that is, an agent with weight α times another must have their share valued or 'weighted' α times the other's, in some manner. If the agents have identical weights they are called *symmetric*, and called *asymmetric* otherwise. The following are the most popular notions defined for the asymmetric setting.

Weighted Nash welfare. Defined in [20] as an extension of the Nash bargaining solution to the asymmetric agents setting, as follows. Note that maximizing the Nash welfare value is identical to the objective of maximizing the sum of logarithms of the agent utility functions over all allocations.

The weighted Nash optimal allocation maximizes the sum of the weighted logarithms of the agent utility values, where the weighted logarithm is the logarithm of the utility value multiplied by the weight of the agent. Equivalently, the weighted Nash welfare value is the weighted geometric mean of the agent utility values, where each utility is raised to a power equal to the agent weight, and agent weights are normalized to have sum equal to 1. An allocation is weighted α-approximately Nash optimal, if it has the weighted Nash welfare value equal to $1/\alpha$ times the maximum such value among all allocations.

Any price share. [21] Recently proposed, the any price share is a notion defined as follows. Suppose the agents weights are normalized to have sum equal to 1. Also suppose the resources have prices associated to them, such that the sum of prices is also normalized to have sum 1. Then an agent with weight b is said to find affordable any subset of the resources, or a *bundle*, of total price at most b. A rational agent, given a price vector, will choose the highest utility bundle among all affordable bundles. Any price share is the utility assured if an adversary sets the prices, and the agent is rational. That is, the lowest possible utility over all price vectors, such that highest utility affordable bundle is chosen for each price vector, is the any price share of the agent.

Weighted envy-freeness and weighted maximin share. Various extensions of the envy-free notions and the maximin share have been proposed for the asymmetric settings. These are beyond the scope of this article. We refer to [22, 23] for discussions of these.

The fair division problem is complete - Given a set of m indivisible resources, n agents with weights and utility functions for the resources, compute a fair allocation of the resources among the agents, according to the desired notion of fairness. We end this section with a discussion of the most popular research directions that are ongoing in this area, with some immediate open problems for each.

1**. New notions that generalize the problem.** From the economic lens, defining new notions of fairness that capture existing and more complex scenarios is a key direction. The current model captures *individual fairness*. One such general setting already being explored is where agents can be strategic, and can collude to form groups. They report false utility functions to accumulate a larger set of resources within the group, and re-compute allocations of these within the subset of agents that colluded. Group fairness is a notion that disincentivizes collusion. Group-wise fair MMS is defined to extend MMS to capture this, and finding the best α for which α-approximate groupwise fair MMS allocations exist is open.

2. **New fairness concepts for current settings.** Every known notion has certain desirable properties as well as drawbacks. For instance, maximin share and EFx are important notions, but they are not known to exist. In fact, maximin fair allocations are proven to not exist. Nash optimal solutions are known to be computationally intractable. While computational research focuses on finding approximately optimal solutions, a caveat is that the approximate guarantees do not have the same compelling reasons to justify they are fair. For instance, α-Nash optimal solutions need not satisfy EF1 or Pareto optimality. Economic research focuses on defining new notions such

that optimal solutions are guaranteed to exist, and whose approximate solutions will also satisfy fairness guarantees.

3. **Existence.** The existence of exactly fair allocations according to some key fairness notions is not known. For instance, it is an open problem to decide if an EFx allocation exists for every fair division instance, even if the instances are restricted to those with 4 agents with linear utility functions. Another example is the α-maximin fair allocation results. Constructive results for particular α values show the existence, while counterexamples negate the existence of β-maximin fair allocations, for $\beta > \alpha$. Closing this gap is an open problem. More specifically for example, for the class of instances where agent utility functions are restricted to be linear, deciding the highest α between $3/4$ and $39/40$ for which an α-MMS allocation always exists is a key open problem.

4. **Design of efficient approximate algorithms.** Without assuming any mathematical properties to the utility functions, it is difficult to achieve good approximation guarantees. For example, suppose one only assumes the functions are monotonically non-decreasing, meaning agents always prefer to have supersets of the set instead of the set, for any set of resources. Then an α-maximin share can be shown to *not exist*, for any $\alpha > 0$. To obtain more meaningful results, we restrict the utility functions to belong to some subclass of functions, which are also meaningful. Assuming linear or additive functions is one such extensively studied setting. There are separate series of works studying the fair division problem with additive function agents, for all the fairness notions, including NSW and MMS. These also show intractability and non-existence results. Resolving the limits of intractability, that is, for NSW and MMS, designing algorithms with improved and tight approximation guarantees is a key direction of research.

5. **Fixed parameter tractability** The fair division problem, even when restricted to linear utility functions, has significant non-existence and intractability results. While searching for the best possible approximation algorithms is important, another direction pursued is fixing some parameters to be constant. For example, settings where the number of agents are constant, or the case of restricting utility functions such that each agent is allowed to have at most k non-zero valued resources, for a constant k, are meaningful and capture several real life settings. Finding better results than the general case for these is an active direction. Perhaps the biggest open problem here is deciding if EFx allocations exist for all instances with 4 agents, as the answer for a general n number of agents has remained elusive after extensive research.

6. **Generalizing other parameters for a fixed fairness notion.** Every fairness notion has been extensively studied for the setting with linear utility functions, when the resources are all goods,

and when the agents have symmetric weights. It is important to initiate the study of efficient algorithm design for more general settings too. There are three key generalizations currently being actively pursued: (i) agents have asymmetric weights, (ii) the resources are chores or mixed, and (iii) the utility functions are more general than linear functions. The important utility functions being explored are budget-additive, separable piece-wise linear and concave (SPLC), matroid rank functions (MRF), weighted matroid rank functions (wMRF), OXS, Submodular, subadditive and XOS functions. The definitions of the classes relevant to us are as follows. Those of the remaining functions can be found in the book [24]. All functions are defined on a set of m indivisible resources, denoted by $[m] = \{1, 2, \cdots, m\}$. We denote each function by $u : 2^{[m]} \to \mathbb{R}_{\geq 0}$.

- Additive or linear: The value or utility of a subset of items is equal to the sum of values of the individual items in the subset. That is, $u(S) = \sum_{j \in S} u(j)$, for all $S \subseteq [m]$.
- OXS: Every function is associated with a complete bipartite graph, denoted by $G(L, R, E)$, where the vertices in the left part L of G each correspond to a distinct resource in $[m]$, and the right part R can have any number of vertices. The value of a subset of resources $S \subseteq [m]$ is equal to the value of a maximum weight matching of the sub-graph of G obtained by restricting the left part L to the vertices that correspond to the resources in S.
- Submodular: Any function that satisfies the property $u(S \cup S') + u(S \cap S') \leq u(S) + u(S')$ is called submodular. An alternate equivalent definition is using the notion of marginal utility values. These are interpreted as the value of an item j over another set of items S, defined as $u(S \cup \{j\}) - u(S)$. Submodular functions are those that satisfy the property of diminishing marginal returns, that is, $u(S \cup \{j\}) - u(S) \geq u(S' \cup \{j\}) - u(S')$, for any sets $S \subseteq S' \subseteq [m]$, and any $j \in [m]$.

These function classes are related to each other as follows.

$$\mathsf{Additive} \subsetneq \mathsf{SPLC} \subsetneq \mathsf{OXS} \subsetneq \mathsf{Submodular} \subsetneq \mathsf{XOS}.$$

$$\mathsf{Additive} \subsetneq \mathsf{BA} \subsetneq \mathsf{Submodular} \subseteq \mathsf{XOS} \subseteq \mathsf{Subadditive},$$

$$\mathsf{MRF} \subsetneq \mathsf{wMRF} \subsetneq \mathsf{Submodular}.$$

Some important open problems here are: (i) finding better than $1/3$-approximate MMS allocations for the fair division problem on goods where the agents have submodular utility functions, and (ii) better than linear factor algorithms for the Nash welfare problem with asymmetric agents, goods, and linear utility functions.

7. **Other tractable relaxations.** Apart from approximation of the optimal solution, it is fruitful to ask if there is some other relaxation of the problem. For instance, a recently initiated line of research finds randomized allocations that ensure ex-post exact envy-free, and ex-ante relaxed envy-free (further relaxed than EF1) guarantees. Such dual guarantees are called *best of both worlds* guarantees, and finding a balance between ex-ante and ex-post, and the correct fairness notions, that ensures strong guarantees on both sides, is an open problem.

8. **Algorithms for guaranteeing multiple notions of fairness.** Every fairness notion has certain drawbacks. Further, most of them do not guarantee economic efficiency. Therefore, an important question is to show the existence of and design algorithms for allocations that (approximately) guarantee multiple fairness and efficiency guarantees. Some important open problems here are, does an EF1+PO allocation exist for agents with submodular utility functions? Design an algorithm for computing a $1/2$-MMS+PO allocation for allocating goods among agents with linear utility functions.

1.2 ROUTING PROBLEMS

Routing games are one of the foundational topics of Algorithmic game theory. Introduced by [25], these games are also popularly studied under the alternate name of congestion games. They have enjoyed enormous attention, both in theory and practice, due to their ability to effectively and efficiently model large and complex networks. Their theory has in turn been influential in shaping AGT, by contributing central notions like the Price of Anarchy, which captures the cost inefficiency of worst case Nash equilibria of games, and regret-minimizing agents. The term *routing games* is mainly used when modeling traffic, an example of a large and complex network. Informally, the task of routing traffic is modeled as a game as follows.

Informal routing game model. Routing games model traffic as follows - the network or a map is specified as a graph where the nodes are locations and edges are direct roads connecting two locations. The players of the game are the users who wish to travel between their specified source and destination nodes. Each player's strategy set is all possible paths between their source and destination nodes. Each road is congestion aware, meaning each user has a cost function associated with every road, that is non-decreasing with respect to the number of users who are assigned this road in the solution. The cost to a player for choosing a particular path or strategy, is the sum of costs of taking the individual edges or roads along the path. A feasible solution is an assignment of a path to every user chosen from their strategy set.

Given this model, a natural question is how to characterize *good* or optimal feasible solutions. There are two main solution concepts - the Nash equilibrium, and the minimum cost solution.

Nash equilibrium. A mixed strategy for each user in a routing game, is a probability distribution over all of their (pure) strategies, that is, paths between their source and destination nodes. A strategy vector is a vector of mixed strategies, one corresponding to each player. A Nash equilibrium of a routing game is a strategy vector where no user has incentive to unilaterally deviate. This means that any other strategy vector, where exactly one user changes their strategy, gives equal or worse cost to this user.

Nash equilibrium solutions have several desirable properties. Among others, a Nash equilibrium is a player's *best response* for the strategies chosen by everyone else. [25] showed that every routing game has a *pure* strategy Nash equilibrium, which means that every player's mixed strategy in the equilibrium solution has probability 1 for exactly one of their (pure) strategies. Moreover, the pure strategy Nash equilibrium can be found by the following simple best response update method - start with an arbitrary feasible solution, and until one reaches an equilibrium, pick an arbitrary player who can switch to a lower cost strategy, and swap their current strategy to their best response strategy.

Optimal cost solution. The solution that minimizes the sum of costs to all users among all feasible solutions, is the optimal cost solution. An optimal cost solution is the most economically efficient solution, and is also Pareto optimal.

The fairness perspective. Routing traffic can be naturally modeled as a fair allocation problem - the roads are the resources, the users are the agents with non-zero utilities for paths between their source and destination nodes. There are two differences in this setting from the indivisible resource allocation problem, as follows. First, the resources are *shareable* that is, the paths assigned to the agents may not be disjoint. And second, the resources have *externalities*, meaning the assignment of one user can affect the cost to other users, because of the congestion aware nature of the network. Several works have separately explored the settings of shareable resources or resources with externalities. Most of them propose fairness notions that extend the concepts of proportionality or envy-freeness here. As envy-freeness is the stronger of these in cake-cutting theory, we ask the questions,

How can envy-related fairness notions be defined in routing settings?

In fair division theory, the main economic considerations while developing any new notion are the following two central axioms - 'equal treatment of equals,' and Pareto optimality. The Pareto optimality requirement seems especially important in a routing setting - to see why, consider the

situation when all the users have the same source and destinations. Assigning them identical routes would be envy-free according to any envy-related definition. This solution would be impractical, as the high congestion will result in high costs to each user. Therefore, our goal is an attempt to define fairness notions that satisfy both requirements. We explore the stronger question,

Are there meaningful notions that ensure envy-freeness and economic efficiency, and are also efficiently computable?

We initiate the study of these questions in our work [5]. Before exploring envy-related concepts, we first discuss why the two popular concepts - Nash equilibrium and optimal cost solutions, may not be the answer to the above question. The optimal cost solutions are Pareto optimal, but are not concerned with individual treatment at all. Nash equilibrium solutions can be considered to be treating equals equally, if fairness is equated to a best response outcome. Nash equilibria may however not be Pareto optimal.

The Price of Anarchy (PoA) results are an argument in favor of using Nash equilibria. PoA is defined as follows. The efficiency of a routing game is the ratio of the total cost of the best Nash equilibrium, which is the equilibrium with the lowest total cost, with the optimal cost of the instance. The PoA of a class of instances is the worst efficiency ratio over all instances. It is known that the PoA of the class of instances where all the congestion cost functions are small degree polynomials is small. For instance, if all cost functions are linear, then any Nash equilibrium has a total cost at most $4/3$ times that of the optimal cost, hence is $4/3$-Pareto optimal, meaning no other solution can give every user $3/4$ times their current cost, and some user a strictly lower than $3/4$ times their current cost. This shows that Nash equilibria are approximately Pareto optimal as well.

However, these ratios seem to rise rapidly with the increase in the degree of the polynomial. Further the famous Braess' paradox example shows that there may exist instances where the Nash equilibria are dominated by feasible solutions that give *every* user a better cost strategy. Therefore, we seek new notions that may be meaningful, and may capture fairness from a different perspective.

In a follow-up work [6], we extend our setting to dynamically changing networks and users appearing in real time in a streaming fashion. This setting is motivated by the dynamically changing travel time estimates of road segments in real world networks. Routing with dynamic costs is an expensive problem in terms of latency, meaning computing the solution in real time for each changed estimate may be infeasible. Therefore, finding the optimal route with respect to the dynamic costs in a practical setting is a difficult task. Further, we also ask that our algorithms ensure fairness guarantees to all users. We propose a new type of user assignment, instead of a single path between their source and destination, and new notions of fairness for this setting.

We make the case that a precomputed set of routes, termed *alternatives*, between two endpoints in the network can be sufficient to nearly match the optimal assignment of users to paths. For each

request, our algorithm evaluates the set of precomputed alternative routes using the dynamic costs, and serves the best one to the user at each round. Our algorithms utilize historical information about the traffic demands in the road network and are semi-online, in the sense that they anticipate a certain amount of traffic, but perform well even when a random subset of the traffic actually appears. We also extend our envy-freeness notions to this *semi-online* setting, where the users cost functions are known apriori, but only a subset of them may appear, in a random order.

CHAPTER 2: LITERATURE REVIEW AND SUMMARY OF RESULTS

2.1 THE NASH WELFARE PROBLEM

The Nash welfare problem is as follows - given a set of m goods, n agents, and utility or valuation functions of the agents for the goods, compute an allocation with the maximum Nash welfare value. To see the mathematical formulation of the problem, let us define some notation.

Notation. Denote by $[m] = \{1, 2, \cdots, m\}$ the resources, by $[n] = \{1, 2, \cdots, n\}$ the agents, and by $A = \{A_1, A_2, \cdots, A_{n-1}, A_n\}$, an allocation or a partition of the resources among the agents, where each agent $i \in [n]$ receives the set A_i. That is, A is any n sized set of subsets of $[m]$ that satisfies $A_i \cap A_k = \emptyset$ for any $i, k \in [n]$, and $\cup_{i\in[n]} A_i = [m]$. Let Π be the set of all possible allocations, that is, $\Pi = \{A = \{A_1, \cdots, A_n\} \mid A_i \cap A_k = \emptyset \forall i, k \in [n], \cup_{i\in[n]} A_i = [m]\}$. Finally, let $v_i : 2^{[m]} \to \mathbb{R}_{\geq 0}$ denote the valuation function of each agent i. If the agents have weights, let the weight of agent i be w_i, such that $\sum_i w_i = 1$.

Nash welfare problem. The problem with symmetric weight agents is to find an allocation in the set $\operatorname{argmax}_{A\in\Pi} \prod_{i\in[n]} v_i(A_i)^{1/n}$. The problem with asymmetric agents is to solve,

$$\operatorname*{argmax}_{A\in\Pi} \prod_{i\in[n]} v_i(A_i)^{w_i}.$$

The Nash welfare objective was shown to satisfy several desirable fairness properties, for example EF1 and PO by [19]. [22] showed that such properties are also satisfied by the asymmetric Nash welfare problem, specifically, PO and a property they call the weak weighted envy-freeness up to one good (wwEF1).

The computational problem is NP-hard, even with 2 symmetric weight agents who have linear valuation functions. A simple reduction from the known NP-hard Partition problem shows this. Therefore, the relaxed version asks to design efficient algorithms to compute an allocation whose Nash welfare value is $1/\alpha$ times that of the maximum Nash welfare value, for an $\alpha > 1$ as large as possible. This approximation problem has been extensively studied in the symmetric agents setting (when every w_i is $1/n$). The most popular works can be divided into two key lines of research - (i) stronger approximation guarantees for the setting where the agents have linear, or as popularly termed *additive* valuations (defined at the end of Section 1.1), and (ii) algorithms for more general cases, with more complex agent valuation functions. We describe these here.

Additive valuations. The symmetric agents problem with additive valuations was shown to be hard to approximate within a factor 1.069 by [26]. On the positive side The first breakthrough result on the Nash welfare problem was [27]. They showed a $2e^{1/e} \approx 2.888$ approximation algorithm, and introduced the notion of *spending restricted market equilibrium*. Later, [28] improved their analysis to achieve an approximation factor 2. An approximation factor 2 algorithm was also shown by [29], who used a different approach, and used the theory of real stable polynomials. The current best factor for the additive functions setting is $e^{1/e} \approx 1.45$, by [30]. Their key idea is a connection between EF1 allocations for agents with identical utility functions, and approximate Nash optimal solutions.

General valuations. Constant factor approximation algorithms were obtained for slight generalizations of the class of additive functions - by [26] for budget-additive functions, by [31] for SPLC functions, and by [32] for a combination of the two. The approximation factor of the last work is $e^{1/e}$, hence matches the best known factor with additive functions.

Our work, [1] initiated the study of this problem for the far larger class of submodular functions. We show an $O(n \log n)$ approximation algorithm for this setting, where n is the number of agents, which additionally allows the agents to have asymmetric weights as well. We also show a 1.58 factor inapproximability for this problem, observing that this setting is strictly harder than the additive setting, where a positive result for a better factor is known. Subsequently, submodular functions and classes closer to these gained the attention of the fair division community. [33, 34] showed linear time algorithms for the larger class of subadditive functions. The inapproximability of the social welfare maximization problem (maximize the sum of values of agents) [35] implies that under the value oracle model, this is the best possible factor [33]. [36] studied the problem for XOS functions, which lie between submodular and subadditive, and gave a sublinear $O(n^{53/54})$ factor algorithm, when both demand and XOS oracles are provided.

Constant factor algorithms for such larger function classes also soon followed, in a series of remarkable works. [37] gave methods to approximate the optimal Nash welfare *value*, but not efficient methods to compute an approximately optimal allocation, within a factor $e^3/(e-1)^2 \approx 6.8$, for a broad class of submodular functions that contains, among others, coverage functions and summations of matroid rank functions. [38] defined and studied the Rado valuations, a subclass of submodular functions that combines the OXS matching constraint with a matroid constraint. They gave an approximation factor of 772 for this setting. By extending their techniques, [39] showed a constant 380 factor approximation algorithm for submodular functions. Recently, [40] showed a $(6+\epsilon)$ factor algorithm for the submodular case, and a $(12+\epsilon)$ factor algorithm that additionally also guarantees the $1/2$-EFx property.

Asymmetric Nash welfare. The asymmetric nash welfare problem was first explored in [41] who showed a $O(m)$ factor algorithm, where m is the number of resources. Our work [1] gave the first algorithms independent of m. We showed an n factor algorithm for the case when the agents have additive utility functions, and an $O(n \log n)$ factor algorithm for the submodular functions setting. [38] studied the problem for the setting with Rado valuations, and gave a $772(w_{max}/w_{min})^3$ factor algorithm, where w_{max} and w_{min} are respectively the largest and smallest entitlements among all agent entitlements. Recently, [40] studied the submodular functions case, by giving a $(w_{max}/w_{avg} + 2 + \epsilon)e$-factor algorithm for this case, where w_{avg} is the average of all agent entitlements.

Tractability for special cases. A different line of work studies special subclasses of the Nash welfare problem with additive utility functions that are tractable, meaning admit an FPTAS or a PTAS. These works are as follows. When the agents have identical additive utility functions, an symmetric weights, [41] show a PTAS that efficiently computes an ϵ-approximate Nash optimal allocation for fixed constant ϵ. [42] gave greedy algorithms that obtained a 1.061 factor guarantees for this case. For the case with constantly many symmetric weight agents, [43] showed an FPTAS. Recently, [44] improved these results to the asymmetric weights setting. Additionally, when the agents have identical k-ary valuations for a constant k, meaning they have at most k different values for the goods, they show an efficient algorithm to obtain the Nash optimal allocation. They also show an efficient algorithm for the case with general monotone non-decreasing functions where each agent has non-zero value for at most 2 goods, and complement by showing that with non-zero value for 3 goods, the problem is NP-hard even with additive functions.

2.1.1 Our results

We initiate the study of the Nash welfare problem for the setting where the agents have asymmetric weights and submodular valuation functions. The submodular functions generalization had not been explored prior to our work, while only a trivial $O(m)$ factor approximation algorithm, where m is the number of items, was known for the asymmetric weights setting. Our results for these settings, and some further interesting properties about our techniques, are as follows.

- SMatch: Our first algorithm is an $O(n)$ factor algorithm, where n is the number of items, for the setting with asymmetric agent weights, and additive valuation functions.
- RepReMatch: Our second algorithm is an $O(n \log n)$ factor algorithm, for the more general setting with submodular valuation functions and asymmetric agent weights.

- Both of our algorithms are simple to understand and involve clever modifications of a greedy repeated matchings approach.
- Both the algorithms ensure the EF1 fairness guarantee, along with the approximate Nash welfare guarantees.
- We also study the special case of the submodular functions, asymmetric weights setting where the number of agents is constant. We resolve this setting, by giving a $(1 - 1/e) \approx 1.5819$ factor algorithm, and a matching hardness of approximation.
- We additionally show that SMatch gives a 1.45 factor guarantee for the setting with restricted additive valuations, meaning that the value for each agent i for any item j is some value v_j or zero. This matches the optimal factor for the setting with general additive valuations and symmetric weight agents.
- We prove the tightness of our analysis by giving counterexamples that show that (i) for slight generalizations of our settings, namely the subadditive and XOS valuations, the same factors cannot be shown by using a similar algorithm, and (ii) for the setting we explore, of additive valuations with asymmetric agent weights, there is an instance which results in an allocation with the proven worst case guarantee.

2.2 THE MAXIMIN SHARE PROBLEM

The maximin share (MMS) value of each agent is the maximum value among all allocations, of the minimum valued bundle in each allocation. That is, using the notation from Section 2.1, the MMS value of an agent with valuation function $v(.)$ is given by,

$$\mathsf{MMS} = \max_{\Pi} \min_{k \in [n]} v(A_i).$$

The Maximin share problem. An allocation A is called MMS-fair, if $v_i(A_i) \geq \mathsf{MMS}_i$, for every agent $i \in [n]$, where $v_i(.)$ is the utility function of agent i, and MMS_i is their MMS value. The maximin share fair allocation problem is, given n agents, m indivisible items, and utility or valuation functions of the agents for the goods, to efficiently compute an MMS-fair allocation.

Note that, by definition of MMS, if the agents have identical valuation functions, their MMS values and the allocations that define these will be identical. Therefore, any allocation that defines their MMS value is MMS-fair. Equivalently, the problem of computing the MMS value of an agent is the same as the problem of computing an MMS-fair allocation for the instance with the same resources but where all the (same number of) agents have a utility function identical to this agent's

function. This problem is NP-hard, as can be shown by a simple reduction from the well known NP-hard Partition problem. However, a PTAS is known [45] since before the problem was defined, from the theory of machine scheduling problems.

With non-identical functions however, [16] showed that MMS-fair allocations may not exist at all. Therefore, the fairness guarantee was relaxed, by searching for approximate MMS allocations, defined as follows. An allocation is called α-MMS-fair, for some $\alpha > 0$, if every agent receives a bundle they value at least α times their MMS value. Before we describe the most popular works on this problem, we observe a key challenge in pursuing this.

Key Challenge. The existence of an α-MMS allocation for the setting with non-identical agents, and additive valuation functions, is not known. Therefore, successful attempts that prove efficient algorithms for larger values of α, also show the important existence guarantee of α-MMS allocations as a corollary.

The α-MMS problem was extensively studied, especially for the setting with additive valuation functions. With additive valuations, the major results can be divided into three further categories, based on whether they explore the fair division of goods, chores, or mixed items.

Goods. [46] was the initial work to study this problem. They showed the existence of MMS allocations in some restricted cases, for example when the value of each good is at most some small value. The first breakthrough result for the general additive function setting was in [16], who showed that MMS allocations may not always exist, but $2/3$-MMS allocations always do. This inspired a series of works that showed progressively simpler methods for the efficient computation of $2/3$-MMS allocations [47, 48, 49]. Then [50] broke the $2/3$ barrier, by showing the existence of a $3/4$-MMS allocation. Finally, the state of the art result was [51], who showed that a $(3/4 + 1/(12n))$-MMS allocation always exists. The PTAS of [45] to compute MMS values can be used to find a $(3/4+1/(12n)-\epsilon)$-MMS allocation for any $\epsilon > 0$ in polynomial time. The same paper [51] also show a strongly polynomial time algorithm to find $3/4$-MMS allocation. Other notable works on the goods only case before being improved by follow-up work are [49, 52, 53, 54]. On the negative side, [55] showed that a better than $39/40$-factor allocation does not exist for all instances in this setting.

Constant number of agents. A small but significant line of work studies the setting with constantly many agents. For three agents [47] showed that a $7/8$-MMS allocation always exists. This factor was later improved to $8/9$ in [56]. For four agents, [50] showed that a $4/5$-MMS allocation always exists. Our work [57] gave an algorithm that, given an instance and any $\epsilon > 0$, computes an $(\alpha - \epsilon)$-MMS allocation, for the highest $\alpha \in (0, 1]$ for which an α-MMS allocation exists for the instance.

Chores. [58] first studied the MMS problem with a chores manna. They introduced an algorithm for finding 2-MMS allocations (Our definition of α-MMS for the mixed manna is consistent for agents with positive as well as negative MMS values. We define α as smaller than 1, and consider $1/\alpha$-MMS valued bundles as α-MMS. Prior results for the chores manna have $\alpha > 1$ and ask for $\alpha \cdot$ MMS valued bundles. We state the approximation factors as defined in the original papers, and ask the reader to invert them when relating with ours). [48] improved the previous result by showing an algorithm for a $4/3$-MMS allocation. Later, [59] improved this result to a $11/9$-MMS allocation. They also showed a PTAS to find $(11/9 + \epsilon)$-MMS allocation and a polynomial time algorithm to find a $5/4$-MMS allocation. [60] improved these results, by introducing the notion of *picking sequences for chores*, and gave a $5/3$-factor algorithm. Recently, [61] gave the best known results. For the more general case of agents with asymmetric weights, they show a 1.733 factor approximation, not only to MMS, but to something called the *chore share,* that is better than another popular share notion called the APS, which itself is known to be better than MMS values. For the symmetric agents case, their algorithm achieves an $8/5$ factor. They also show a negative result, that a better than $3/2$-factor algorithm is not tractable for asymptotically large values of n, where n is the number of agents. One the negative side, [55] show that a better than $44/43$-MMS allocation does not exist for all instances.

Mixed. Our work [57] inititated the study of the problem for a mixed resources setting. We first showed that for any $\alpha > 0$, there are instances where an α-MMS allocation does not exist! Therefore, one cannot pursue the problem for the mixed case in the same flavor as the goods or chores cases. We ask and explore the bext possible question - is there an efficient algorithm, that, given an instance, computes the best α-MMS allocation that exists for the instance. This means that, if one is given an instance with mixed items but with agents that have identical valuation functions, then the algorithm should return a 1-MMS allocation. The main result of our work is a PTAS for this problem, for a particular setting, and complement the result by showing that any generalization of this setting makes the problem inapproximable to any positive value. Another work [2], also studies this problem, but in a different manner. Instead of studying the most general setting which can be efficiently solved, they argue that the examples that imply the harsh inapproximability of the problem all occur from the MMS values becoming close to zero. Therefore they study the setting where one is guaranteed that the MMS value is bounded away from zero, and show a PTAS for the identical agents case.

General valuation functions. [62] initiated the study of this problem for submodular valuations and gave an efficient algorithm for 0.21-MMS. [50] showed the existence of and a PTAS for computing $1/3$-MMS allocations with submodular valuations, and a non-existence of $3/4$-MMS

allocations, with submodular valuations; the existence of a $1/5$-MMS, efficient computation of a $1/8$-MMS, and the non-existence of a better than $1/2$-MMS allocation with XOS valuations; and finally the existence of a $1/10\lceil \log m \rceil$-MMS allocation, where m is the number of items, with subadditive valuations. [37] improved the result to a 0.3666-MMS allocation for a special case of XOS functions called the *hereditary set systems,* which essentially are XOS functions with weighted binary marginal values. For subadditive valuations, [63] showed this existence is equivalent to an existence of a $1/(n \log n)$-MMS allocation, where n is thus number of agents, and further improved it to a $1/(\log n \log \log n)$-MMS allocation existence. [64] showed exact MMS+PO allocations for submodular valuation functions with binary marginals. Recently, our work [3] gave a $1/3(1 + (2/3)/(n - (2/3))) > 1/3$-factor algorithm for the problem with OXS valuation functions, and a $2/3$-factor inapproximability. The best known factors for this setting, which lies between additive and submodular classes, was $1/3$-factor algorithm due to the known results for the submodular case, and a $39/40$-factor non-existence due to the additive case.

Extensions to other settings. The MMS problem has been studied extensively under various other models in the goods only setting - for instance, with asymmetric agents [52], group fairness [65, 66], in matroids [56], with additional constraints [56, 67], for agents with externalities [68, 69], with graph constraints [70, 71], and with strategic agents [72]. In the chores only setting too, variations of the weighted MMS [73] and asymmetric agents [74] notions have been investigated.

2.2.1 Our results

Mixed Manna

We initiate the study of finding MMS + PO allocations for a mixed manna.

We first show that, for any fixed $\alpha \in (0, 1]$, an α-MMS allocation may not always exist. This rules out efficient computation for any fixed α, and naturally raises the following problem.

Design an efficient algorithm to find an α-MMS + PO *allocation for the best possible* α*, i.e., the maximum* $\alpha \in (0, 1]$ *for which it exists.*

This *exact* problem is intractable: In the case of identical agents, an $(\alpha = 1)$-MMS allocation exists by definition. However, finding one is known to be NP-hard even for a goods manna. Also, checking if a given instance admits an MMS allocation is known to be in $\mathsf{NP}^{\mathsf{NP}}$, but not known to be in NP ([46]). On the positive side, a polynomial-time approximation scheme (PTAS) is known for this case due to [45]; given a *constant* $\epsilon \in (0, 1]$, the algorithm finds a $(1 - \epsilon)$-MMS allocation in polynomial time. No such result is known when the agents are not identical. Guaranteeing PO

in addition adds to the complexity, since even checking if a given allocation is PO is coNP-hard even with two identical agents ([75]). In light of these results, we ask,

Question. *Can we design a* PTAS*, namely an efficient algorithm to find an* $(\alpha - \epsilon)$-MMS $+ \gamma$-PO *allocation, given* $\epsilon, \gamma > 0$*, for the best possible* α*?*

Our Contribution. We show the following dichotomy result: We derive two conditions and show that the problem is tractable under these conditions, while dropping either renders the problem intractable. The two conditions are: (i) number of agents n is a constant, and (ii) for every agent i, her total (absolute) value for all the items ($|v_i(\mathcal{M})|$) is significantly greater than the minimum of her total value of goods (v_i^+) and her total (absolute) value for chores (v_i^-), i.e., for a constant $\tau > 0$, $|v_i(\mathcal{M})| \geq \tau \cdot \min\{v_i^+, v_i^-\}$.

In particular, first, for instances satisfying (i) and (ii), we design a PTAS (as asked in the above question). Second, we show that if either condition is not satisfied, then finding an α-MMS allocation for *any* $\alpha \in (0, 1]$ is NP-hard, even with *identical agents* where a solution exists for $\alpha = 1$. This hardness is striking because it shows inapproximability within *any* non-trivial factor when either (i) or (ii) is not satisfied. This also indicates that the two conditions are unavoidable.

Our algorithm, in principle, gives a little more than a PTAS. It runs in time $2^{O(1/\min\{\epsilon^2,\gamma^2\})}poly(\mathrm{m})$ for given $\epsilon,\ \gamma$, thus gives polynomial run-time for ϵ, γ as small as $O(1/\sqrt{\log m})$, where $m = |\mathcal{M}|$.

α-MMS$+$PO *for goods (chores) manna.* As a corollary, we obtain a PTAS for finding α-MMS$+$ PO allocations of a goods manna and a chores manna when the number of agents is a constant. This improves the previous results for these settings in two aspects: (i) provides the best possible approximation factor; factors better than the general case known for good manna are $4/5$ for $n = 4$ by [50], $8/9$ for $n = 3$ by [56], and 1 for $n = 2$ by [46], and (ii) provides an additional (approximate) PO guarantee.

Challenges. The key challenge in solving this question is handling items of high value to any agent. In the goods or chores mannas, these items can be greedily assigned, for example as singleton bundles. But in a mixed manna, *high valued* goods (chores) may have to be bundled with specific sets of chores (goods) or low valued items to form lesser valued bundles. Secondly, the MMS values of the agents, and the α for which α-MMS allocation exist, both are not known. In fact, computing the exact MMS values is NP-hard (even with a goods manna).

PTAS *to find* MMS *values.* As the first key step for our main algorithm, we design a PTAS that returns $(1 - \epsilon)$ approximate MMS values of agents, which may be of independent interest.

A new technique to prove PO. Since certifying a PO allocation is a coNP-hard problem ([75]), known works (e.g., [30, 76, 77]) maintain a PO allocation with market equilibrium as a certificate . We develop a novel approach to ensure PO with α-MMS through LP rounding. The LP itself is intuitive, however the rounding is involved. It makes use of *envy-graph* and properties of the MMS in a novel way.

OXS valuations

We initiate the study of maximin share fairness problem for the setting where the agents have OXS valuation functions. These are the first studies on OXS functions in fair division, to the best of our knowledge. We will discuss the following results in Chapter 5.

Efficient algorithm. We show the existence of $\frac{1}{3}(1 + \frac{2/3}{(n-2/3)})$-MMS allocation, breaking the barrier of $1/3$ for the OXS valuations, and design a PTAS to compute one. As a corollary, it ensures much better factors for small number of agents, *e.g.,* $1/2$ with 2 agents, $3/7$ with 3 agents, $2/5$ with 4 agents and so on. To break the barrier of $1/3$ we uncover important properties of the OXS functions (w.r.t. the MMS problem). Importantly, we show that, every agent can assign one *representative value* to every good and as a result there is an ordering of the goods. Using this, we analyze the round-robin procedure to obtain a factor better than $1/3$. The challenge in the analysis is to bound the loss in value when a good is matched to an agent, as $O(n)$ goods get discarded due to this matching, due to being connected to the same right side vertex as the matched good in the OXS graph. These insights, together with a simple algorithm for beyond-additive valuations, may be of independent interest to analyze other fairness notions for OXS and other special classes of submodular functions, like gross-substitutes.

Non-existence of better than $2/3$-MMS. We show a simple example with 2 agents and 4 goods where an allocation strictly better than $2/3$-MMS does not exist. This gives an improved non-existence result for valuations that subsume OXS, namely *gross-substitutes*, *Rado*, and *submodular* valuations, as the previous best known result was for the submodular functions [50], with a non-existence of $3/4$-MMS allocations.

Computing MMS value. We show that the problem of computing MMS values of agents with OXS functions is strongly NP-hard, negating the possibility of a PTAS. To counter the impossibility result, we show an efficient algorithm to compute the MMS value of an agent within a factor of $1/2$.

EF1 +PO and EF1+MSW allocations with identical agents. A key subroutine of our algorithm with identical agents resolves another popular fair and efficient notion, namely the EF1+PO allocation. An allocation is called EF1 if every agent values their bundle more than any other agent's bundle upon removing some good from the other bundle. An allocation is called PO if there is no other allocation where every agent receives a bundle of equal or higher value, and at least one agent gets a strictly better bundle. Finding an EF1+PO allocation is a widely studied problem, with little success (the end of this section has a brief overview), and the existence of such an allocation is open, even with identical agents in the beyond-additive valuations setting.

We show such an allocation exists in the OXS valuations setting with identical agents. In fact, we show the existence of an allocation that is not only PO but also has the maximum social welfare (MSW), meaning the sum of values of all the bundles is maximized.

2.3 ROUTING

The problem we investigate, is to define notions of economic efficiency and fairness for a routing or congestion game setting, and explore their computability. While we initiate the study of envy-related notions, to the best of our knowledge, there are a few works studying other fairness related notions.

Fairness in congestion settings. Chakrabarty et al. in [78] study certain aspects of fairness and optimality in a singleton congestion game, which corresponds to a routing game on parallel links. Their work also includes a result on network games, which we improve in our work. Correa et al. in [79] study the problem of minimizing the maximum cost in the Wardrop (i.e., non-atomic flow) model of traffic. Kleinberg et al. in [80] study *max-min* fairness in bandwidth allocation. Max-min fairness requires that no agent can be improved without harming some other agent. Meyers and Schulz in [81] present several hardness results on optimizing cost in congestion games, which is related to minimizing travel time in atomic routing games, and identify several special cases that are solvable in polynomial time.

Several previous works focus on optimizing travel time with no emphasis on fairness, typically in the context of Stackelberg routing, where a subset of the flow is routed centrally (e.g., from a navigation platform) and the remaining traffic responds selfishly. In Roughgarden [82], it was shown that the Stackelberg routing problem is NP-Hard (even in these simple graphs) and that two algorithms, called Scale and Largest Latency First, achieve constant factor approximations. The analysis was generalized to arbitrary networks by Swamy in [83], which obtained a constant factor approximation for the case of general networks and polynomial delay functions, and also improved bounds for the case of parallel links. The bounds for polynomials were improved in [84] by Bonifaci et al. In terms of data-driven studies, Colak et al [85] and Youn et al. [86] study the inefficiency due to selfish behavior and other parameters in multiple cities. The work in [87] combines the design of Stackelberg algorithms and data-driven evaluation.

Alternates. Computing alternative routes has been often applied as a tool in routing work that focuses on road networks. Early work in the theory literature, such as the k-shortest path problem [88, 89] can be considered a precursor to this line of work. The work in [90, 91] introduce the use of alternative routes for the purposes of satisfying user preferences that go beyond minimizing the travel time. We point the reader to Kobitzsch's thesis [92], which contains a comprehensive description of the literature on alternative route computation, but also mention the state of the art on the topic: The works in [93, 94] give the best performing version of the *penalty method* [95] and the *plateau method* [96, 97]. More recent work studies alternative routes in the context of fast shortest path computation [98] and robustness against changes in the road network and in user preferences

[99]. Our study differs from the above papers in that we use alternative route computation to find near optimal solutions for all navigation requests, taking the delays induced by the resulting traffic congestion into account.

2.3.1 Our results

Envy-ratio. We define *Envy-freeness* for a fair routing instance as follows. For every routing assignment, the *envy ratio of user* i, denoted by $\mathsf{Er_i}$, is the ratio of the cost of the path assigned to i to the least cost among all assigned paths, according to their cost function. The *envy ratio of a routing assignment* is defined is the maximum envy between any pair of users, that is $\mathsf{Er} = \max_i \mathsf{Er_i}$. A routing assignment is called fair or envy-free (EF1) if its envy ratio is 1.

We define α approximately envy-free or α-EF1 routing assignments as those with Er at most α. Intuitively, users must prefer routing assignments with low envy ratio.

Optimal cost solution. We denote by OPT, a solution with the lowest total cost to all the users, and define γ approximately optimal or γ-OPT routing assignments as those with total cost at most γ times the cost of an optimal solution's cost.

We explore the following problem.

Definition 2.3.1 (α-EF1 + γ-OPT.)**.** *Given a fair routing instance and constants* $\alpha, \gamma \geq 1$*, find an* α-EF1 *and* γ-OPT *routing assignment.*

Our results are as follows.

- Non-existence: we begin with an impossibility result. We show that for certain networks, it is not possible to achieve optimality and no envy at the same time.
- Intractability: we proceed with a hardness result, showing that it is NP-Hard to decide whether an optimal solution that is as fair as possible exists and that similarly it is NP-Hard to decide whether a near-optimal and fair solution exists.
- 3-EF1 + OPT exists: On the positive side, we first show that for a single origin-destination pair network, even with users with preferences, there exists an assignment of routes that is optimal in terms of total travel time and every driver has envy at most 3, meaning that there can't be a route of another driver that has cost less than $1/3$ times her own.
- 4-EF1 + 2-OPT algorithm: Next we show that for the same setting, we can efficiently find a routing assignment that has total travel time at most twice the optimal travel time, and every driver has envy at most 4.

- 3-EF1 + OPT algorithm with identical agents: Finally, we show that for the previously studied setting with users with identical preferences, we can find a routing assignment that is optimal and has envy strictly less than 3.

Extending our setting to assign a set of paths, termed *alternates*, to each agent, we model the problem and its approximation parameters as follows.

Instance. We are given the network graph G, two nodes s, t, in G, and n users who wish to travel from s to t. C is the set of (identical) non-negative monotone cost functions of the users for each edge $c_e : n \to \mathbb{R}_{\geq 0}$ for each edge e in the road network G.

Alternates assignment algorithm. A k-ALTALGO is an algorithm that assigns $O(k)$ $s-t$ paths to each user in the routing instance. We will assume that once the set of alternates is specified, the paths in real time are generated by a greedy congestion aware shortest path computation algorithm, termed ROUTFrmALT. That is, ROUTFrmALT is an algorithm that given an instance, a subset of cn users and a set of alternate routes to each user, arbitrarily orders the users, and assigns to each user in order the shortest congestion aware path, meaning the shortest path according to the cost functions computed using the current congestion values.

Optimal algorithm. We compare the output of ROUTFrmALT with a greedy shortest path computation algorithm that would compute these paths in real time, with access to the entire network information, instead of any subset of k paths in it.

We define a notion that will be useful in defining the fairness and efficiency notions. The *rank* of any $s-t$ path is the iteration number in which it was assigned to some user by the optimal algorithm.

Fairness and efficiency. The fairness of our algorithm is measured using two notions, each defined for a different stage of the algorithm. The first is again the *envy ratio,* defined as the ratio between the worst and the best routes of any route assignment. If the envy ratio of the assignment produced by ROUTFrmALT is β, then we say its corresponding k-ALTALGO is a β-envy-free algorithm.

Our second notion measures the fairness of the assignment of alternate routes as follows. Consider the alternate routes assigned to any user i. We say this user envies some other user j up to at most p paths, denoted by EFp (read as *envy-free up to p paths*) if after removing their commonly assigned paths, the best path of i is no worse than the p^{th} best path of j. We call a k-ALTALGO EFp if it generates an assignment that is EFp for each user.

We measure efficiency of the i^{th} user, by comparing the rank of the path assigned by the algorithm, with i, which is the rank of the path that the i^{th} user was assigned in the optimal algorithm. The efficiency of the algorithm is the worst rank among all users. That is, if the ROUTFrmALT that uses a k-ALTALGO results in all the users choosing paths ranked at most $(\alpha \cdot cn)$ for *any* cn-sized subset of the users in expectation, then we say that this k-ALTALGO is an α-rank approximation of the optimal algorithm.

Note that the higher the value of k, easier it is to ensure a good rank approximation guarantee. For example, if $k = n$, then we can assign all paths to every user, and any subset of cn users will be able to choose one of the top cn paths. We show such an assignment exists with high probability for a particular ordering of the users for $k = \log n$.

Our main results are two algorithms with the following fairness and efficiency guarantees.

- A $\log$ n-ALTALGO deterministic algorithm which gives a constant factor rank guarantee in expectation, and is 2-envy-free and EF1.
- Then with a slightly higher number of paths and using randomization, we obtain O$(\log^2 \mathsf{n}/\epsilon)$-ALTALGO, which gives a PTAS like $(1+\epsilon)$-rank guarantee with high probability, and is 2-envy-free.

CHAPTER 3: NASH WELFARE

In this chapter, we study the problem of approximating the maximum Nash social welfare (NSW) when allocating a set $\mathcal{M}$ of m indivisible items among a set $\mathcal{A}$ of n agents with non-negative monotone *submodular* valuations $v_i : 2^{\mathcal{M}} \to \mathbb{R}_+$, and unequal or *asymmetric* entitlements called *agent weights*. Let $\Pi_n(\mathcal{M})$ denote the set of all allocations, i.e., $\{(\mathbf{x}_1, \ldots, \mathbf{x}_n) \mid \cup_i \mathbf{x}_i = \mathcal{M};\ \mathbf{x}_i \cap \mathbf{x}_j = \emptyset, \forall i \neq j\}$. The NSW problem is to find an allocation maximizing the following weighted geometric mean of valuations,

$$\underset{(\mathbf{x}_1,\ldots,\mathbf{x}_n)\in\Pi_n(\mathcal{M})}{\operatorname{argmax}} \left(\prod_{i\in\mathcal{A}} v_i(\mathbf{x}_i)^{\eta_i}\right)^{1/\sum_{i\in\mathcal{A}}\eta_i},$$

where η_i is the weight of agent i. We call this the *Asymmetric Submodular* NSW *problem*. [1] When agents are symmetric, $\eta_i = 1, \forall i \in \mathcal{A}$.

The asymmetric NSW and the submodular NSW problems were first raised in [28, 30]. The main results of this chapter are two approximation algorithms, SMatch and RepReMatch for asymmetric agents with additive and submodular valuations respectively. Both algorithms are simple to understand and involve non-trivial modifications of a greedy repeated matchings approach. Allocations of high valued items are done separately by un-matching certain items and re-matching them, by processes that are different in both algorithms. We show that these approaches achieve approximation factors of $O(n)$ and $O(n \log n)$ for additive and submodular case respectively, which is independent of the number of items. For additive valuations, our algorithm outputs an allocation that is also EF1.

Model

We formally define the valuation functions considered, and their relations to other popular functions. For convenience, we also use $v_i(j)$ instead of $v_i(\{j\})$ to denote the valuation of agent i for item j.

1. Additive: Given valuation $v_i(j)$ of each agent i for every item j, the valuation for a set of items is the sum of the individual valuations. That is, $\forall \mathcal{S} \subseteq \mathcal{M}, v_i(\mathcal{S}) = \sum_{j\in\mathcal{S}} v_i(j)$.

2. Monotone Submodular: Let $v_i(\mathcal{S}_1 \mid \mathcal{S}_2)$ denote the marginal utility of agent i for a set $\mathcal{S}_1$ of items over set $\mathcal{S}_2$, where $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{M}$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. Then, the valuation function

[1] We refer to various special cases of the problem as the α μ NSW problem, where α is the nature of agents, symmetric or asymmetric, and μ is the type of agent valuation functions. We skip one or both qualifiers when they are clear from the context.

of every agent is a monotonically non decreasing function $v_i : 2^{\mathcal{M}} \to \mathbb{R}_+$ that satisfies the submodularity constraint that for all $i \in \mathcal{A}, h \in \mathcal{M}, \mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{M}$,

$$v_i(h \mid \mathcal{S}_1 \cup \mathcal{S}_2) \leq v_i(h \mid \mathcal{S}_1).$$

Results

Table 3.1 summarizes approximation guarantees of the algorithms RepReMatch and SMatch under popular valuation functions [24], formally defined in Section 3. All current best known results are also stated here for reference.

Valuations	**Symmetric Agents**		**Asymmetric Agents**	
	Hardness	**Algorithm**	**Hardness**	**Algorithm**
Restricted Additive	1.069 [26]	1.45[30] [S]	1.069 [26]	$O(n)$ [S]
Additive	1.069 [26]	1.45 [30]	1.069 [26]	$O(n)$ [S]
Budget-additive	—"—	1.45 [32]	—"—	—"—
SPLC	—"—	—"—	—"—	$O(n \log n)$ [R]
OXS Gross substitutes	—"—	$O(n \log n)$ [R]	—"—	—"—
Submodular	1.5819 [Thm 3.2.1]	—"—	1.5819 [Thm 3.2.1]	—"—
XOS Subadditive	—"—	$O(m)$ [41]	—"—	$O(m)$ [41]

Table 3.1: Summary of results. Every entry has the best known approximation guarantee for the setting followed by the reference, from this paper or otherwise, that establishes it. Here, [S] and [R] respectively refer to Algorithms SMatch and RepReMatch.

To complement these results, we also provide a 1.5819-factor hardness of approximation result for the submodular NSW problem in Section 3.2. This hardness even applies to the case when the number of agents is constant. This shows that the general problem is strictly harder than the settings studied so far, for which 1.45 factor approximation algorithms are known.

For the special case of the submodular NSW problem where the number of agents is constant, we describe another algorithm with a *matching* 1.5819 *approximation factor* in Section 3.3, hence resolving this case completely. Finally in the same section, we show that for the symmetric additive NSW problem, the allocation of items returned by SMatch also satisfies EF1. Finally, a 1.45-factor guarantee can be shown for the further special case of restricted additive valuations, by showing that the allocation returned by the algorithm in this case is PO. This matches the current best known approximation factor for this case.

Techniques

Algorithms RepReMatch **and** SMatch**.** The main idea used in these algorithms is shown in Lemma 3.1.1 in Section 3.1, which we restate in informal terms here.

Lemma (informal)**.** *For $k = O(n)$ and for every agent i, after removing a set S_i of k items that minimizes i's valuation for the remaining items, repeatedly matching the remaining items $\mathcal{M}\backslash(\cup_i S_i)$ to locally maximize the* NSW *objective gives every agent an allocation of value at least a $1/n$ fraction of her valuation for the remaining set of items, i.e, $v_i(\mathcal{M}\backslash(\cup_i S_i))/n$.*

When the valuation functions are additive, then such a set of k items can be efficiently found. SMatch then follows by combining these results. As a natural extension (See Remark 3.4.1), we note that these arguments also work for the slightly generalized case of budget-additive valuations.

It is known from [100] that finding a set of minimum valuation with lower bounded cardinality for monotone submodular functions is inapproximable within $\sqrt{m/\ln m}$ factor, where m is the number of items. Hence, the above lemma by itself is insufficient for the submodular NSW problem. For this case, we prove Lemma 3.1.3 in Section 3.1, that implies the following statement.

Lemma (informal)**.** *When items are iteratively matched to agents to locally maximize the weighted geometric mean of agents' valuations from their matched items, then the set of items allocated in the first $\log n + 1$ matchings if re-matched have a matching with every agent getting an item she values at least as much as the highest valued item from her* NSW *optimizing allocation.*

RepReMatch combines the two results in a repeated matching, un-matching and then re-matching algorithm, by applying Lemma 3.1.1 to the set of items that remain unallocated after a phase of $\log n + 1$ matchings.

An observation used while designing both algorithms is that maximizing the logarithm of the NSW function instead of the NSW objective does not change the optimal allocation(s). Doing so allows us to maximize the (weighted) sum of the logarithms of individual agent valuations, instead of the (weighted) product of valuations. Hence, the edge weights defined in the various graphs for computing the matchings in both SMatch and RepReMatch are logarithms of agent valuations for some allocations.

Submodular NSW **with constant number of agents.** This is a different approach that uses techniques of maximizing submodular functions over matroids developed in [101], and a reduction of fair division problems to the problem of maximizing a submodular function over matroids from [102]. At a high level, we first maximize the continuous relaxations of agent valuation functions, then round them using a randomized algorithm to obtain an integral allocation of items. The two key results used in designing the algorithm are Theorems 3.3.2 and 3.3.3.

Hardness of approximation. The submodular ALLOCATION problem is to maximize the sum of valuations of agents over integral allocations of items. [103] describe a reduction of MAX-3-COLORING, which is NP-Hard to approximate within a constant factor, to ALLOCATION. We prove that this reduction also establishes the same hardness for the submodular NSW problem. **Organization.** In Section 3.4, we describe the algorithm SMatch and analysis for the additive NSW problem. In Section 3.1, we present the algorithm for submodular valuations, RepReMatch. Next, Section 3.2 contains the hardness proof for the submodular setting. Finally, Section 3.3 presents the results for the special cases of submodular NSW with constant number of agents, symmetric additive NSW, and symmetric additive NSW with restricted valuations. Finally, in Section 3.5 we present counter examples to prove tightness of the analysis of Algorithms RepReMatch and SMatch.

3.1 SUBMODULAR VALUATIONS

In this section we present the algorithm RepReMatch, given in Algorithm 1, for approximating the NSW objective under submodular valuations. We will prove the following relation between the NSW of the allocation $\mathbf{x}$ returned by RepReMatch and the optimal geometric mean OPT.

Theorem 3.1.1. $\mathsf{NSW}(\mathbf{x}) \geq \frac{1}{2\mathsf{n}(\log \mathsf{n}+2)}\mathsf{OPT}$.

3.1.1 Algorithm

RepReMatch takes as input an instance of the NSW problem, denoted by $(\mathcal{A}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$, where $\mathcal{A}$ is the set of agents, $\mathcal{M}$ is the set of items, and $(v_i)_{i\in\mathcal{N}} = \{v_1, v_2 \ldots, v_n\}$ is the set of agents' monotone submodular valuation functions, and generates an allocation vector $\mathbf{x}$. Each agent $i \in \mathcal{A}$ is associated with a positive weight η_i.

RepReMatch runs in three phases. In the first phase, in every iteration, we define a weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{M}^{rem}, \mathcal{W})$ as follows. $\mathcal{M}^{rem}$ is the set of items that are still unallocated ($\mathcal{M}^{rem} = \mathcal{M}$ initially). The weight of edge $(i, j), i \in \mathcal{A}, j \in \mathcal{M}^{rem}$, denoted by $w(i, j) \in \mathcal{W}$, is defined as the logarithm of the valuation of the agent for the singleton set having this item, scaled by the agent's weight. That is, $w(i, j) = \eta_i \log(v_i(j))$. We then compute a maximum weight matching in this graph, and allocate to agents the items they were matched to (if any). This process is repeated for $\log n + 1$ iterations.

We perform a similar repeated matching process in the second phase, with different edge weight definitions for the graphs Γ. We start this phase by assigning empty bundles to all agents. Here, the weight of an edge between agent i and item j is defined as the logarithm of the valuation of agent i for the set of items currently allocated to her in Phase 2 of RepReMatch, scaled by her weight.

That is, if we denote the items allocated in t iterations of Phase 2 as $\mathbf{x}^2_{i,t}$, in $(t+1)^{st}$ iteration, $w(i,j) = \eta_i \log(v_i(\mathbf{x}^2_{i,t} \cup \{j\}))$.

In the final phase, we re-match the items allocated in Phase 1. We release these items from their agents, and define $\mathcal{M}^{rem}$ as union of these items. We define Γ by letting the edge weights reflect the total valuation of the agent upon receiving the corresponding item, i.e. $w(i,j) = \eta_i \log(v_i(\mathbf{x}^2_i \cup \{j\}))$, where $\mathbf{x}^2_i$ is the final set of items allocated to i in Phase 2. We compute one maximum weight matching for Γ so defined, and allocate all items along the matched edges. All remaining items are then arbitrarily allocated. The final allocations to all agents, denoted as $\mathbf{x} = \{\mathbf{x}_i\}_{i\in\mathcal{A}}$, is the output of RepReMatch.

3.1.2 Notation

There are three phases in RepReMatch. We denote the set of items received by agent i in Phase $p \in \{1,2,3\}$ by $\mathbf{x}^p_i$, and its size $|\mathbf{x}^p_i|$ by τ^p_i. Similarly, $\mathbf{x}_i$ and τ_i respectively denote the final set of items received by agent i and the size of this set. Note that Phase 3 releases and re-allocates selected items of Phase 1, thus τ_i is not equal to $\tau^1_i + \tau^2_i + \tau^3_i$. The items allocated to the agents in Phase 2 are denoted by $\mathbf{x}^2_i = \{h^1_i, h^2_i \ldots, h^{\tau^2_i}_i\}$. We also refer to the complete set of items received in iterations 1 to t of Phase p by $\mathbf{x}^p_{i,t}$, for any $p \in \{1,2,3\}$.

For the analysis, the marginal utility of an agent i for an item j over a set of items $\mathcal{S}$ is denoted by $v_i(j \mid \mathcal{S}) = v_i(\{j\} \cup \mathcal{S}) - v_i(\mathcal{S})$. Similarly, we denote by $v_i(\mathcal{S}_1 \mid \mathcal{S}_2)$ the marginal utility of set $\mathcal{S}_1$ of items over set $\mathcal{S}_2$ where $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{M}$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. We use $\mathbf{x}^* = \{\mathbf{x}^*_i \mid i \in \mathcal{A}\}$ to denote the optimal allocation of all items that maximizes the NSW, and τ^*_i for $|\mathbf{x}^*_i|$. For every agent i, items in $\mathbf{x}^*_i$ are ranked so that g^j_i is the item that gives i the highest marginal utility over all higher ranked items. That is, for $j = 1$, g^1_i is the item that gives i the highest marginal utility over $\emptyset$, and for all $2 \le j \le \tau^*_i$, $g^j_i = \operatorname{argmax}_{g \in \mathbf{x}^*_i \setminus \{g^1_i, \ldots, g^{j-1}_i\}} v_i(g \mid \{g^1_i, \ldots, g^{j-1}_i\})$.[2]

We let $\bar{\mathbf{x}}^*_i$ be the set of items from $\mathbf{x}^*_i$ that are not allocated (to any agent) at the end of Phase 1, and denote by $\bar{v}^*_i = v_i(\bar{\mathbf{x}}^*_i)$ and $\bar{\tau}^*_i = |\bar{\mathbf{x}}^*_i|$ respectively the total valuation and number of these items. For readability, to specify the valuation for a set of items $\mathcal{S}_1 = \{s^1_1, \ldots s^{k_1}_1\}$, instead of $v_i(\{s^1_1, \ldots, s^{k_1}_1\})$, we also use $v_i(s^1_1, \ldots, s^{k_1}_1)$. Similarly, while defining the marginal utility of a set $\mathcal{S}_2 = \{s^1_2, \ldots, s^{k_2}_2\}$ over $\mathcal{S}_1$ instead of writing $v_i(\{s^1_2, \ldots, s^{k_2}_2\} \mid \{s^1_1, \ldots, s^{k_1}_1\})$, we also use $v_i(s^1_2, \ldots, s^{k_2}_2 \mid s^1_1, \ldots, s^{k_1}_1)$.

[2]Since the valuations are monotone submodular, this ensures that $v_i(g^j_i \mid \{g^1_i, \ldots, g^{j-1}_i\}) \ge v_i(g^k_i \mid \{g^1_i, \ldots, g^{k-1}_i\})$ for all $k \ge j$. This implies that in any subset of ℓ items in the optimal bundle, the highest ranked item's marginal contribution is at least $1/\ell$ times that of this set, when the marginal contribution is counted in this way.

Algorithm 1: RepReMatch for the Asymmetric Submodular NSW problem

Input : A set $\mathcal{A}$ of n agents with weights $\eta_i,\ \forall i \in \mathcal{A}$, a set $\mathcal{M}$ of m indivisible items, and valuations $v_i : 2^{\mathcal{M}} \to \mathbb{R}_+$, where $v_i(\mathcal{S})$ is the valuation of agent $i \in \mathcal{A}$ for a set of items $\mathcal{S} \subseteq \mathcal{M}$.

Output: An allocation that approximately optimizes the NSW objective

Phase 1:

$\mathbf{x}_i^1 \leftarrow \emptyset,\ \forall i \in \mathcal{A}$ // $\mathbf{x}_i^1$'s store the set of items allocated in Phase 1

$\mathcal{G}^{rem} \leftarrow \mathcal{M}$ // set of unallocated items before every iteration

$t \leftarrow 0$ // iteration counter

while $\mathcal{M}^{rem} \neq \emptyset$ *and* $t \leq \log n + 1$ **do**

  Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{M}^{rem}, \mathcal{W})$ with weights $\mathcal{W} = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(j)), \forall i \in \mathcal{A}, j \in \mathcal{M}^{rem}\}$

  Compute a maximum weight matching $\mathcal{M}$ for Γ

  $\mathbf{x}_i^1 \leftarrow \mathbf{x}_i^1 \cup \{j\}, \quad \forall (i,j) \in \mathcal{M}$ // allocate items to agents according to $\mathcal{M}$

  $\mathcal{M}^{rem} \leftarrow \mathcal{M}^{rem} \backslash \{j \mid (i,j) \in \mathcal{M}\};\ t \leftarrow t+1$ // remove allocated items

end

Phase 2:

For all i, $\mathbf{x}_i^2 \leftarrow \emptyset$ // $\mathbf{x}_i^2$'s are the sets of items allocated in Phase 2

while $\mathcal{G}^{rem} \neq \emptyset$ **do**

  Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{M}^{rem}, \mathcal{W})$ with weights $W = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(\mathbf{x}_{i,t}^2 \cup \{j\})), \forall i \in \mathcal{A}, j \in \mathcal{M}^{rem}\}$

  Compute a maximum weight matching $\mathcal{M}$ for Γ

  $\mathbf{x}_i^2 \leftarrow \mathbf{x}_i^2 \cup \{j\}, \quad \forall (i,j) \in \mathcal{M}$ // allocate items to agents according to $\mathcal{M}$

  $\mathcal{M}^{rem} \leftarrow \mathcal{M}^{rem} \backslash \{j \mid (i,j) \in \mathcal{M}\}$ // remove allocated items

end

Phase 3:

$\mathcal{M}^{rem} \leftarrow \bigcup_i \mathbf{x}_i^1$ // release items allocated in Phase 1

Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{M}^{rem}, \mathcal{W})$ with $W = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(\mathbf{x}_i^2 \cup \{j\})), \forall i \in \mathcal{A}, j \in \mathcal{M}^{rem}\}$

Compute a maximum weight matching $\mathcal{M}$ for Γ

$\mathbf{x}_i^2 \leftarrow \mathbf{x}_i^2 \cup \{j\}, \quad \forall (i,j) \in \mathcal{M}$ // allocate items to agents according to $\mathcal{M}$

Arbitrarily allocate rest of the items to agents, let $\mathbf{x} = \{\mathbf{x}_i\}_{i \in \mathcal{A}}$ denote the final allocation

return $\mathbf{x}$

3.1.3 Analysis

We will prove Theorem 3.1.1 using a series of supporting lemmas. We first prove that in Phase 2, the minimum marginal utility of an item allocated to an agent over her current allocation from previous iterations of Phase 2 is not too small. This is the main result that allows us to bound the minimum valuation of the set of items allocated in Phase 2.

In the t^{th} iteration of Phase 2, RepReMatch finds a maximum weight matching. Here the algorithm tries to assign to each agent an item that gives her the maximum marginal utility over her currently allocated set of items. However, every agent is competing with $n-1$ other agents to get this item. So, instead of receiving the best item, she might lose a few high ranked items to other agents. Consider the intersection of the set of items that agent i loses to other agents in the t^{th} iteration with the set of items left from her optimal bundle at the beginning of t^{th} iteration. We will refer to this set of items by $\mathcal{S}_i^t$. Let the number of items in $\mathcal{S}_i^t$ be k_i^t.

For the analysis of RepReMatch, we also introduce the notion of *attainable items* for every iteration. $\mathcal{S}_i^t$ is the set of an agent's preferred items that she lost to other agents. The items that are now left are referred as the set of *attainable* items of the agent. Note that in any matching, every agent gets an item equivalent to her best attainable item, that is, an item for which her marginal valuation (over her current allocation) is at least equal to that from her highest marginally valued attainable item.

For all i, we denote the intersection of the set of *attainable* items in the t^{th} iteration and agent i's optimal bundle $\mathbf{x}_i^*$ by $\bar{\mathbf{x}}_{i,t}^*$, and let $u_i^* = v_i(\bar{\mathbf{x}}_{i,1}^*) = v_i(\bar{\mathbf{x}}_i^* \setminus \mathcal{S}_i^1)$ be the total valuation of *attainable* items at first iteration of Phase 2. In the following lemma, we prove a lower bound on the marginal valuation of the set of *attainable* items over the set of items that the algorithm has already allocated to the agent.

Lemma 3.1.1. *For any $j \in [\tau_i^2 - 1]$,*

$$v_i(\bar{\mathbf{x}}_{i,j+1}^* | h_i^1, \ldots, h_i^j) \;\geq\; u_i^* \;-\; k_i^2 v_i(h_i^1) \;-\; \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \;\mid\; h_i^1, \ldots, h_i^{t-1}) \;-\; v_i(h_i^1, h_i^2 \ldots, h_i^j).$$

Proof. We prove this lemma using induction on the number of iterations t. Consider the base case when $t = 2$. Agent i has already been allocated h_i^1. She now has at most $\bar{\tau}_i^* - k_i^1$ items left from $\bar{\mathbf{x}}_i^*$ that are not yet allocated. In the next iteration the agent loses k_i^2 items to other agents and receives h_i^2. Each of the remaining $\bar{\tau}_i^* - k_i^1$ items have marginal utility at most $v_i(h_i^1)$ over $\emptyset$. Thus, the marginal utility of these items over h_i^1 is also at most $v_i(h_i^1)$. We bound the total marginal valuation of $\bar{\mathbf{x}}_{i,2}^*$ over $\{h_i^1\}$, by considering two cases.
Case 1: $h_i^1 \notin \bar{\mathbf{x}}_{i,1}^*$: By monotonicity of v_i, $v_i(\bar{\mathbf{x}}_{i,2}^* \mid h_i^1) \geq v_i(\bar{\mathbf{x}}_{i,2}^*) - v_i(h_i^1) = v_i(\bar{\mathbf{x}}_{i,1}^* \setminus \mathcal{S}_i^2) - v_i(h_i^1)$.
Case 2: $h_i^1 \in \bar{\mathbf{x}}_{i,1}^*$: Here, $v_i(\bar{\mathbf{x}}_{i,2}^* \mid h_i^1) = v_i(\bar{\mathbf{x}}_{i,2}^* \cup \{h_i^1\}) - v_i(h_i^1) = v_i(\bar{\mathbf{x}}_{i,1}^* \setminus \mathcal{S}_i^2) - v_i(h_i^1)$.

In both cases, submodularity of valuations and the fact that for all $j \in \mathcal{S}_i^2, v_i(j) \leq v_i(h_i^1)$ implies,

$$\begin{aligned} v_i(\bar{\mathbf{x}}_{i,2}^* \mid h_i^1) &\geq v_i(\bar{\mathbf{x}}_{i,1}^*) - v_i(\mathcal{S}_i^2) - v_i(h_i^1) \\ &\geq u_i^* - k_i^2 v_i(h_i^1) - v_i(h_i^1), \end{aligned}$$

proving the base case. Now assume the lemma is true for all $t \leq r$ iterations, for some r, i.e.,

$$v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^{r-1}) \geq u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^{r-1}).$$

Consider the $(r+1)^{st}$ iteration. Again, we analyze two cases. **Case 1:**[$h_i^r \notin \bar{\mathbf{x}}_{i,r}^*$:]

$$\begin{aligned} v_i(\bar{\mathbf{x}}_{i,r+1}^* \mid h_i^1, \ldots, h_i^r) &= v_i(\bar{\mathbf{x}}_{i,r}^* \setminus \mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^r) \\ &\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^r) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^r) \\ &\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^r) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}) \\ &\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}). \end{aligned}$$

The submodularity of v_i gives the first two inequalities, and monotonicity of v_i implies the last. **Case 2:** [$h_i^r \in \bar{\mathbf{x}}_{i,r}^*$:]

$$\begin{aligned} v_i(\bar{\mathbf{x}}_{i,r+1}^* \mid h_i^1, \ldots, h_i^r) &= v_i(\bar{\mathbf{x}}_{i,r+1}^* \cup \{h_i^r\} \mid h_i^1, \ldots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) \\ &= v_i(\bar{\mathbf{x}}_{i,r}^* \setminus \mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) \\ &\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}). \end{aligned}$$

Here the second expression follows as $\bar{\mathbf{x}}_{i,r}^* = \bar{\mathbf{x}}_{i,r+1}^* \cup \{h_i^r\} \cup \mathcal{S}_i^{r+1}$, and the last follows from the definition of submodularity of the valuations. In both cases, from the induction hypothesis we get,

$$\begin{aligned} v_i(\bar{\mathbf{x}}_{i,r+1}^* \mid h_i^1, \ldots, h_i^r) \geq u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^{r-1}) \\ - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}). \end{aligned}$$

Finally, since RepReMatch assigns the item with highest marginal utility from the set of *attainable*

items, and each item in $\mathcal{S}_i^{r+1}$ is attainable at r^{th} iteration,

$$\begin{aligned} v_i(\bar{\mathbf{x}}^*_{i,r+1} \mid h_i^1, \ldots, h_i^r) \geq\ & u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^{r-1}) \\ & - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) - k_i^{r+1} v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) \\ =\ & u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^r). \end{aligned}$$

QED.

The above lemma directly allows us to give a lower bound on the marginal valuation of item received by the agent in $(j+1)^{th}$ iteration over the items received in previous iterations. We state and prove this in the following corollary.

Corollary 3.1.1. *For any* $j \in [\tau_i^2 - 1]$,

$$\begin{aligned} v_i(h_i^{j+1} \mid h_i^1, \ldots, h_i^j) \geq \frac{1}{\bar{\tau}_i^* - \sum_{t=1}^{j+1} k_i^t} \Bigg(& u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) \\ & - v_i(h_i^1, h_i^2 \ldots, h_i^j) \Bigg). \end{aligned}$$

Proof. In any setting with a set of items $\mathcal{S} = \{s_1, \ldots s_k\}$, and a monotone submodular valuation v on this set, if $v(\mathcal{S}) = u$, then there exists an item $s \in \mathcal{S}$ such that $v(s) \geq u/k$. Thus, with $\mathcal{S} = \bar{\mathbf{x}}^*_{i,j+1}$, $k = \bar{\tau}_i^* - \sum_{t=1}^{j+1} k_i^t$, for the submodular valuation function $v_i(\cdot \mid \{h_i^1, \ldots, h_i^j\})$, we can say that at iteration $j+1$, h_i^{j+1} will have a marginal valuation at least,

$$\frac{1}{\bar{\tau}_i^* - \sum_{t=1}^{j+1} k_i^t} v_i(\bar{\mathbf{x}}^*_{i,j+1} | h_i^1, \ldots, h_i^j).$$

Together with Lemma 3.1.1, this proves the corollary. Note that at any iteration t, if the received item h_i^t is from $\bar{\mathbf{x}}^*_{i,t}$, then the denominator reduces further by 1, and the bound still holds. QED.

In the following lemma, we give a lower bound on the total valuation of the items received by the agent in Phase 2.

Lemma 3.1.2. $v_i(h_i^1, \ldots, h_i^{\tau_i^2}) \geq \frac{u_i^*}{n}$.

Proof. Recall that u_i^* is the valuation of the items from $\bar{\mathbf{x}}_i^*$ *after* she loses items in $\mathcal{S}_i^1$ to other agents in the first iteration of Phase 2 and $\bar{\tau}_i^*$ is the number of items in $\bar{\mathbf{x}}_i^*$. From Corollary 3.1.1,

total valuation of the items obtained by agent i in Phase 2 is bounded as follows.

$$v_i(h_i^1, \dots h_i^{\tau_i^2}) = v_i(h_i^1, \dots, h_i^{\tau_i^2-1}) + v_i(h_i^{\tau_i^2} \mid h_i^1, \dots, h_i^{\tau_i^2-1})$$
$$\Rightarrow v_i(h_i^1, \dots h_i^{\tau_i^2}) \geq v_i(h_i^1, \dots, h_i^{\tau_i^2-1}) + \frac{1}{\bar{\tau_i}^* - \sum_{t=0}^{\tau_i^2-1} k_i^{t+1}} \Bigg(u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{\tau_i^2-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \dots h_i^{t-1}) - v_i(h_i^1, h_i^2 \dots, h_i^{\tau_i^2-1}) \Bigg).$$

By definition, τ_i^2 is the last iteration of Phase 2 in which agent i gets matched to some item. After this iteration, at most n items from her optimal bundle remain unallocated, else she would have received one more item in the $(\tau_i^2+1)^{st}$ iteration. This means the optimal number of items $\bar{\tau_i}^* - \sum_{t=0}^{\tau_i^2-1} k_i^{t+1} \leq n$, hence the denominator of the second term in the above equation is at most n. Again, we note here that if at any iteration t, the item assigned to agent i was from $\bar{\mathbf{x}}_{i,t}^*$, then the denominator will be further reduced by 1 for all such iterations, and the inequality still remains true when k_i^t is replaced by $k_i^t + 1$. Combined with the fact that an agent can lose at most $n-1$ items in every iteration, we get $k_i^t \leq n-1$, implying,

$$\begin{aligned}
v_i(h_i^1, \dots h_i^{\tau_i^2}) &\geq v_i(h_i^1, \dots, h_i^{\tau_i^2-1}) + \frac{1}{n} \Bigg(u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{\tau_i^2-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \dots h_i^{t-1}) \\
&\qquad - v_i(h_i^1, h_i^2 \dots, h_i^{\tau_i^2-1}) \Bigg) \\
&\geq v_i(h_i^1, \dots, h_i^{\tau_i^2-1}) + \frac{1}{n} \Bigg(u_i^* - (n-1) v_i(h_i^1) - \sum_{t=2}^{\tau_i^2-1} (n-1) v_i(h_i^t \mid h_i^1, \dots h_i^{t-1}) \\
&\qquad - v_i(h_i^1, h_i^2 \dots, h_i^{\tau_i^2-1}) \Bigg) \\
&= v_i(h_i^1, \dots, h_i^{\tau_i^2-1}) + \frac{1}{n} (u_i^* - (n-1) v_i(h_i^1, h_i^2 \dots, h_i^{\tau_i^2-1}) - v_i(h_i^1, h_i^2 \dots, h_i^{\tau_i^2-1})) \\
&= \frac{u_i^*}{n}.
\end{aligned}$$

QED.

Remark 3.1.1. *In Lemma 3.1.1 and its subsequent Corollary 3.1.1 and Lemma 3.1.2, if* $u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \mid h_i^1, \dots, h_i^{t-1}) - v_i(h_i^1, \dots, h_i^j)$ *becomes negative for any* $j \in [\tau_i^2 - 1]$,

then we have

$$\begin{aligned} u_i^* &\leq k_i^2 v_i(h_i^1) + \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) + v_i(h_i^1, \ldots, h_i^j) \\ &\leq (n-1) v_i(h_i^1) + \sum_{t=2}^{j} (n-1) v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) + v_i(h_i^1, \ldots, h_i^j) \\ &= n \cdot v_i(h_1, \ldots, h_i^j) \leq n \cdot v_i(h_1, \ldots, h_i^{\tau_i^2 - 1}), \end{aligned}$$

which implies that Lemma 3.1.2 holds.

We now bound the minimum valuation that can be obtained by every agent in Phase 3. Recall that g_1^i is the item that gives the highest marginal utility over the empty set to agent i. Before proceeding, we define

$$\mathcal{M}_i^1 := \{g \in \mathcal{M} \mid v_i(g \mid \emptyset) \geq v_i(g_i^1 \mid \emptyset)\}.$$

Lemma 3.1.3. *Consider the complete bipartite graph where the set of agents $\mathcal{A}$, and the set of items allocated in the first Phase of* RepReMatch *are the parts, and edge weights are the weighted logarithm of the agent's valuation for the bundle of items containing the item adjacent to the edge and items allocated in Phase 2. That is, consider $\Gamma(\mathcal{A}, \mathcal{M} = \bigcup_i \mathbf{x}_i^1, \mathcal{W} = \{w(i,j) = \eta_i \log(v_i(\{j\} \cup \mathbf{x}_i^2))\})$. In this graph, there exists a matching where each agent i gets matched to an item from their highest valued set of items $\mathcal{M}_i^1$.*

Proof. Among all feasible matchings between the set of agents and the set of items released after t iterations of Phase 1, consider the set of matchings $\mathcal{M}$ where all the agents whose entire $\mathcal{M}_i^1$ is in this set of items are matched to some item from their $\mathcal{M}_i^1$s. Arbitrarily pick a matching from a subset of this set of matchings where maximum number of agents are matched to some item from their $\mathcal{M}_i^1$. Denote this matching by $\mathcal{M}^t$. Note that as for every set of agents $\mathcal{S}$ we have $|\bigcup_{i \in \mathcal{S}} \mathcal{M}_i^1| \geq |\mathcal{S}|$, in $\mathcal{M}^t$, the set of agents not matched to an item from their $\mathcal{M}_i^1$ each have at least one item from this set still unallocated after t iterations.

Let $\mathcal{A}_t$ denote the set of agents that are not matched to any item from their $\mathcal{M}_i^1$ in $\mathcal{M}^t$. We prove by induction on t that $|\mathcal{A}_t| \leq n/2^t$.

For the base case of the induction, when $t = 1$, we count the number of agents who did not receive any item from their own $\mathcal{M}_i^1$ in the maximum weight matching of the algorithm. We know that before the first iteration, every item is unallocated. An agent will not receive any item from $\mathcal{M}_i^1$ only if all items from this set are allocated to other agents in the matching. Hence, if α agents did not receive any item from their $\mathcal{M}_i^1$, all items from at least α number of $\mathcal{M}_i^1$ sets got matched to some agent(s) in the first matching. If $\alpha < n/2$, then more than $n/2$ agents themselves received

some item from their $\mathcal{M}_i^1$. If $\alpha \geq n/2$, then at least α items, each from a different $\mathcal{M}_i^1$ were allocated. In either case, releasing the allocation of the first matching releases at least $n/2$ items, each belonging in a distinct agent's $\mathcal{M}_i^1$. Hence, in $\mathcal{M}^1$ at least $n/2$ agents receive an item from their $\mathcal{M}_i^1$, and $|\mathcal{A}_1| \leq n/2$.

For the inductive step, we assume the claim is true for the first t iterations. That is, for every $k \leq t$, in $\mathcal{M}^k$, at most $n/2^k$ agents do not receive an item from their $\mathcal{M}_i^1$'s.

Before the $(t+1)^{st}$ iteration begins, we know that for every agent in $\mathcal{A}_t$, at least one item from their $\mathcal{M}_i^1$ is still unallocated. Again by the reasoning of the base case, at least half of the agents in $\mathcal{A}_t$ will have some item from their $\mathcal{M}_i^1$ allocated in the $(t+1)^{st}$ matching (possibly to some other agent). Hence, in $\mathcal{M}^{(t+1)}$, $|\mathcal{A}_{(t+1)}| \leq |\mathcal{A}_t|/2$. By the inductive hypothesis, $|\mathcal{A}_{(t+1)}| \leq n/2^{(t+1)}$. QED.

Proof of Theorem 3.1.1. From Lemma 3.1.2,

$$v_i(h_i^1, \ldots, h_i^{\tau_i^2}) \geq \frac{u_i^*}{n}.$$

By Lemma 3.1.3, giving each agent her own g_i^1 or some item, denoted by say h_i^{1*}, that gives her a marginal utility over $\emptyset$ at least as much as $v_i(g_i^1)$ is a feasible matching before Phase 3 begins. Therefore, we get,

$$\mathsf{NSW}(\mathbf{x}) \geq \left(\prod_{i=1}^{n} (v_i(h_i^{1*}, h_i^2, \ldots, h_i^{\tau_i^2}))^{\eta_i} \right)^{1/(\sum_{i=1}^n \eta_i)}. \tag{3.1}$$

Since the valuation functions are monotonic,

$$v_i(h_i^{1*}, h_i^2, \ldots, h_i^{\tau_i^2}) \geq v_i(h_i^{1*}) \geq v_i(g_i^1).$$

Phase 1 of the algorithm runs for $\log n + 1$ iterations and each iteration allocates n items. Thus $|\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*| \leq n(\log n + 1)$ and $|\mathcal{S}_i^1| \leq n$ implying, $|(\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*) \cup \mathcal{S}_i^1| \leq n(\log n + 2)$. Thus,

$$v_i(g_i^1) \geq \frac{1}{n(\log n + 2)} v_i((\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*) \cup \mathcal{S}_i^1).$$

Also,

$$v_i(h_i^{1*}, h_i^1, \ldots, h_i^{\tau_i^2}) \geq v_i(h_i^1, \ldots, h_i^{\tau_i^2}) \geq \frac{u_i^*}{n} = \frac{1}{n} v_i(\bar{\mathbf{x}}_i^* \setminus \mathcal{S}_i^1).$$

Thus,

$$\begin{aligned}
&v_i(h_i^{1*}, h_i^1, \ldots, h_i^{\tau_i^2}) \\
&\geq \frac{1}{2}\left(\frac{1}{n(\log n+2)} v_i((\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*) \cup \mathcal{S}_i^1) + \frac{1}{n} v_i(\bar{\mathbf{x}}_i^* \setminus \mathcal{S}_i^1)\right) \\
&\geq \frac{1}{2}\frac{1}{n(\log n+2)} v_i(((\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*) \cup \mathcal{S}_i^1) \cup (\bar{\mathbf{x}}_i^* \setminus \mathcal{S}_i^1)) \\
&= \frac{1}{2}\frac{1}{n(\log n+2)} v_i(\mathbf{x}_i^*).
\end{aligned}$$

The second inequality follows from the submodularity of valuations. The last bound, together with (3.1) gives,

$$\begin{aligned}
\mathsf{NSW}(\mathbf{x}) &\geq \left(\prod_{i=1}^{n}\left(\frac{1}{2}\frac{1}{n(\log n+2)} v_i(\mathbf{x}_i^*)\right)^{\eta_i}\right)^{1/\sum_{i=1}^n \eta_i} \\
&\geq \frac{1}{2}\frac{1}{n(\log n+2)}\mathsf{OPT}. \qquad \text{QED.}
\end{aligned}$$

Remark 3.1.2. *We would like to point out that even if Phases* 1 *and* 2 *perform some kind of repeated matchings, the edge weight definitions make them very different. In the proof of Lemma 3.1.3, we require that a maximum weight matching matches agents to items according to agent valuations for the single item. That is, in all iterations of Phase* 1*, the edge weights of the graph in the future are the valuation of the agent for the set containing the single item, and not the increase in the agent's valuation upon adding this item to her current allocation. These quantities are different when the valuations are submodular. For lower bounding the valuation from the lower ranked items, we need to consider the marginal increase, as defined in Phase* 2*. But Lemma 3.1.3 may not hold true if marginal increase in valuations is considered for the initial iterations, hence Phase* 1 *is required.*

3.2 HARDNESS OF APPROXIMATION

We complement our results for the submodular case with a $\frac{e}{(e-1)}$-factor hardness of approximation. Formally, we prove the following theorem.

Theorem 3.2.1. *Unless* $\mathsf{P} = \mathsf{NP}$*, there is no polynomial time* $\frac{e}{(e-1)}$*-factor approximation algorithm for the submodular* NSW *problem, even when agents are symmetric and have identical valuations.*

Proof. We show this using the hardness of approximation result of the ALLOCATION problem proved in [103]. We first summarize the relevant parts of [103]. The ALLOCATION problem is to

find an allocation of a set of indivisible items among a set of agents with monotone submodular utilities for the items, such that the sum of the utilities of all agents is maximized. Note that if the valuation functions were additive, the problem is trivial, and an optimal allocation gives every item to the agent who values it the most. To obtain a hardness of approximation result for the submodular case, the MAX-3-COLORING problem is reduced to the ALLOCATION problem. MAX-3-COLORING, the problem of determining what fraction of edges of a graph can be properly colored when 3 colors are used to colors all vertices of the graph, is known to be NP-Hard to approximate within some constant factor c. The reduction from MAX-3-COLORING generates an instance of ALLOCATION with symmetric agents having identical submodular valuation functions for the items. The reduction is such that for instances of MAX-3-COLORING with optimal value 1, the corresponding ALLOCATION instance has an optimal value of nV, where n is the number of agents in the instance, and V is a function of the input parameters of MAX-3-COLORING. In this case, every agent receives a set of items of utility V. For instances of MAX-3-COLORING with optimal value at most c, it is shown that the optimal sum of utilities of the resulting ALLOCATION instance cannot be higher than $(1 - 1/\mathrm{e})nV$.

For proving hardness of the submodular NSW problem, we note that the input of the ALLOCATION and NSW problems are the same. So let us consider the instance generated by the reduction as that of an NSW maximizing problem. From the results of [103], we can prove the following claims.

- If the optimal value of MAX-3-COLORING is 1, then the NSW of the reduced instance is V. As every agent receives a set of items of value V, the NSW is also V.
- If the optimal value of MAX-3-COLORING is at most c, then the NSW is at most $(1-1/\mathrm{e})V$. Applying the AM-GM inequality establishes that the NSW is at most $1/n$ times the sum of utilities, which is proven to be at most $(1 - 1/\mathrm{e})nV$.

As MAX-3-COLORING cannot be approximated within a factor c, thus NSW of a problem with submodular utilities cannot be approximated within a factor $\frac{\mathrm{e}}{(\mathrm{e}-1)}$.

As the ALLOCATION problem now considered as an NSW problem had symmetric agents and identical submodular valuation functions, the NSW problem also satisfies these properties. QED.

3.3 SPECIAL CASES

3.3.1 Submodular NSW with Constant Number of Agents

In this section, we describe a constant factor algorithm for a special case of the submodular NSW problem. Specifically, we prove the following theorem.

Theorem 3.3.1. *For any constant $\epsilon > 0$ and a constant number of agents $n \geq 2$, there is a $(1 - 1/e - \epsilon)$-factor approximation algorithm for the* NSW *problem with monotone submodular valuations, in the value oracle model. Additionally, this is the best possible factor independent of n, and any factor better than $(1 - (1 - 1/n)^n + \epsilon)$ would require exponentially many queries, unless* P = NP.

The key results that establish this result are from the theory of submodular function maximization developed in [101]. The broad approach for approximately maximizing a discrete monotone submodular function is to optimize a popular continuous relaxation of the same, called the multilinear extension, and round the solution using a randomized rounding scheme. We will use an algorithm that approximately maximizes multiple discrete submodular functions, described in [101], as the main subroutine of our algorithm for the submodular NSW problem, hence first we give an overview of it, starting with a definition of the multilinear extension.

Definition 3.3.1 (Multilinear Extension of a submodular function)**.** *: Given a discrete submodular function $f : 2^m \to \mathbb{R}_+$, its multilinear extension $F : [0,1]^m \to \mathbb{R}_+$, at a point $y \in [0,1]^m$, is defined as the expected value of $f(z)$ at a point $z \in \{0,1\}^m$ obtained by rounding y such that each coordinate y_i is rounded to 1 with probability y_i, and to 0 otherwise. That is,*

$$F(y) = \mathbb{E}[f(z)] = \sum_{X \subseteq [m]} f(X) \prod_{i \in X} y_i \prod_{i \notin X} (1 - y_i).$$

The following theorem proves that the multilinear extensions of multiple discrete submodular functions defined over a matroid polytope can be simultaneously approximated to optimal values within constant factors.

Theorem 3.3.2. *[101] Consider monotone submodular functions $f_1, \ldots, f_n : 2^N \to \mathbb{R}_+$, their multilinear extensions $F_i : [0,1]^N \to \mathbb{R}_+$ and a matroid polytope $P \subseteq [0,1]^N$. There is a polynomial time algorithm which, given $V_1, ..., V_n \in \mathbb{R}_+$, either finds a point $x \in P$ such that $F_i(x) \geq (1 - 1/e)V_i$ for each i, or returns a certificate that there is no point $x \in P$ such that $F_i(x) \geq V_i$ for all i.*

Given a discrete monotone submodular function f defined over a matroid, a rounding scheme called the *swap rounding* algorithm can be applied to round a solution of its multilinear extension to a feasible point in the domain of f, which is an independent set of the matroid. At a high level, in the rounding scheme, it is first shown that every solution of the multilinear extension can be expressed as a convex combination of independent sets such that for any two sets S_0 and S_1 in the convex combination, there is at least one element in each set that is not present in the other, that is $\exists e_0 \in S_0 \backslash S_1$ and $\exists e_1 \in S_1 \backslash S_0$. The rounding method then iteratively merges two arbitrarily

chosen sets S_0 and S_1 into one new set as follows. Until both sets are not the same, one set S_i is randomly chosen with probability proportional to the coefficient of its original version in the convex combination β_i, that is S_i is chosen with probability $\beta_i/(\beta_0+\beta_1)$, and altered by removing e_i from it and adding e_{1-i}. The coefficient of the new set obtained by this merge process is the sum of those of the sets merged, i.e, $\beta_0+\beta_1$.

The following lower tail bound proves that with high probability, the loss in the function value by swap rounding is not too much.

Theorem 3.3.3. *[101] Let $f:\{0,1\}^n \to \mathbb{R}_+$ be a monotone submodular function with marginal values in $[0,1]$, and $F:[0,1]^n \to \mathbb{R}_+$ its multilinear extension. Let $(x_1,...,x_n) \in P(M)$ be a point in a matroid polytope and $(X_1,...,X_n) \in \{0,1\}^n$ a random solution obtained from it by randomized swap rounding. Let $\mu_0 = F(x_1,...,x_n)$ and $\delta > 0$. Then*

$$Pr[f(X_1,...,X_n) \leq (1-\delta)\mu_0] \leq e^{-\mu_0\delta^2/8}.$$

In short, for a matroid $M(X,I)$, given monotone submodular functions $f_i:\{0,1\}^m \to \mathbb{R}_+, i \in [n]$ over the matroid polytope, and values $v_i, i \in [n]$, there is an efficient algorithm that determines if there is an independent set $S \in I$ such that $f_i(S) \geq (1-1/e)v_i$ for every i.

To use this algorithm to solve the submodular NSW problem, we define a matroid $M(X,I)$ as follows. This construction was first described in [104], and also used for approximating the submodular welfare in [102]. From the sets of agents $\mathcal{A}$ and items $\mathcal{M}$, we define the ground set $X = \mathcal{A} \times \mathcal{M}$. The independent sets are all feasible integral allocations $I = \{S \subseteq X \mid \forall j : |S \cap \{\mathcal{A} \times \{j\}\}| \leq 1\}$. The valuation functions of every agent $u_i : \{0,1\}^m \to \mathbb{R}_+$ translate naturally to submodular functions over this matroid $f_i : I \to \mathbb{R}_+$, with $f_i(S) = u_i(\mathcal{M}_i)$, where $\mathcal{M}_i = \{j \in \mathcal{M} \mid (i,j) \in S\}$. With this construction, for any set of values $V_i, i \in [n]$, checking if there is an integral allocation of items that gives valuations at least (approximately) V_i to each agent i is equivalent to checking if there is an independent set in this matroid that has value V_i for every agent i.

The algorithm for approximating the NSW is now straightforward, and given in Algorithm 2. Essentially, we guess the optimal NSW value OPT, and the utility of every agent in the optimal allocation V_i, and check if there is an allocation X that gives every agent i a bundle of value at least (approximately) V_i. As every agent can receive at most Max utility, Max is a trivial upper bound for the maximum value of NSW, hence we perform a binary search for the optimal value in the range $(0, Max]$. Searching for sets V_i by enumerating only those sets with values that are powers of $(1+\delta)$ for some constant $\delta > 0$ will reduce the time complexity of the algorithm to $O(poly(\log(Max)/\delta))$ instead of $O(poly(Max))$, by changing the approximation factor to $(1-1/e)(1-\delta) \leq (1-1/e-\epsilon)$ for some $\epsilon > 0$.

Algorithm 2: Approximate the Submodular NSW with constant number of agents

Input : A set $\mathcal{A}$ of n agents with weights $\eta_i,\ \forall i \in \mathcal{A}$, a set $\mathcal{M}$ of m indivisible items, and monotone submodular valuations $u_i : 2^{\mathcal{M}} \to \mathbb{R}_+$.

Output: An allocation that approximates the NSW.

For any value $Max > 0$ that is a power of $(1+\delta)$, scale all valuation functions such that $u_i(\mathcal{M}) = Max$ for all i. // Max is an upper bound on NSW objective

$OPT = Max$ // OPT is the optimal NSW objective

define $\beta > 0,\ \delta > 0$ as small positive constants

while $OPT \leq Max$ **do**

  flag=0

  for *any set in* $V = \{[V_1, V_2, \ldots, V_n] \mid \prod_i V_i = OPT, \forall i : V_i = (1+\delta)^{k_i}\ for\ some\ k_i\}$ **do**

    if *there is an allocation* $\mathbf{x}$ *of* $\mathcal{M}$ *such that* $u_i(\mathbf{x}_i) \geq (1-1/e)V_i$ *for all* i **then**

      $\mathbf{x}^* = \mathbf{x}, flag = 1$ // $flag = 1$ if current OPT value is feasible

    end

  end

  if *flag=1* **then**

    $OPT = OPT + (Max + \beta - OPT)/2$ // search if a higher value is also feasible. Adding β ensures $OPT > Max$ finally, and algorithm converges

  else

    $Max = OPT, OPT = OPT/2$ // search for a lower feasible value

  end

  $OPT =$ nearest power of $(1+\delta)$ greater than OPT

  $Max =$ nearest power of $(1+\delta)$ greater than Max

end

return $\mathbf{x}^*$

The hardness claim in Theorem 3.3.1 follows from the proof of Theorem 3.2.1. It was shown that in the case where the optimal value of the MAX-3-COLORING instance was 1, every agent in the reduced NSW instance received a bundle of items of value V, else the total NSW could not be more than $(1-(1-1/n)^n)V$.

3.3.2 Symmetric Additive NSW

We now prove that SMatch gives an allocation that also satisfies the EF1 property, making it not only approximately efficient but also a fair allocation. EF1 is formally defined as follows.

Definition 3.3.2 ([4])**.** *Envy-Free up to one item (*EF1*): An allocation* $\mathbf{x}$ *of* m *indivisible items among* n *agents satisfies the envy-free up to one item property, if for any pair of agents* $i, \hat{i}$*, either* $v_i(\mathbf{x}_i) \geq v_i(\mathbf{x}_{\hat{i}})$*, or there exists some item* $g \in \mathbf{x}_{\hat{i}}$ *such that* $v_i(\mathbf{x}_i) \geq v_i(\mathbf{x}_{\hat{i}} \backslash \{g\})$.

That is, if an agent i values another agent $\hat{i}$'s allocation more than her own, which is termed commonly by saying agent i *envies* agent $\hat{i}$, then there must be some item in $\hat{i}$'s allocation upon whose removal this envy is eliminated.

Theorem 3.3.4. *The output of* SMatch *satisfies the* EF1 *fairness property.*

Proof. For every agent i and $j \geq 1$, the item h_j^i allocated to i in the j^{th} iteration of SMatch is valued more by i than all items $h_k^{i'}$, $k > j$ allocated to any other agent i' in the future iterations, as otherwise i would have been matched to the other higher valued item in the j^{th} matching. Hence, $\sum_{t=1}^{j} v_i(h_t^i) \geq \sum_{t=2}^{j} v_i(h_t^{i'})$. That is, after removing the first item $h_1^{i'}$ from any agent's bundle, the sum of valuations of the remaining items for agent i is not higher than her current total valuation. Thus, after removing the item allocated to any agent in the first matching, agent i does not envy the remaining bundle, making the allocation EF1. QED.

Remark 3.3.1. *In fact, the same proof implies that our algorithm satisfies the strong* EF1 *property, defined in [105]. Intuitively, an allocation satisfies the strong* EF1 *property if upon removing the same item, every agent does not envy an agent's allocation any more. Formally,*

Definition 3.3.3 (Strong EF1:)**.** *An allocation* $\mathbf{x}$ *satisfies strong* EF1 *if for every agent* $i \in \mathcal{A}$*, there exists an item* $g_i \in \mathbf{x}_i$ *such that no other agent envies the set* $\mathbf{x}_i \backslash \{g_i\}$*, i.e.,* $\forall j \in \mathcal{A},\ v_j(\mathbf{x}_j) \geq v_j(\mathbf{x}_i \backslash \{g_i\})$.

3.3.3 Symmetric Restricted Additive NSW

For the special case when the valuations are restricted, meaning the valuation of any item v_{ij} is either some value v_j or 0, we now prove SMatch gives a constant factor approximation to the optimal NSW.

Theorem 3.3.5. SMatch *solves the symmetric* NSW *problem for restricted additive valuations within a factor* 1.45 *of the optimal.*

Proof. We prove that x^*, the allocation returned by SMatch, is Pareto Optimal (PO). Combined with the statement of Theorem 3.3.4, and a result of [30] which proves that any allocation that satisfies both EF1 and PO approximates NSW with symmetric, additive valuations within a 1.45 factor, we get the required result. An allocation of items x is called Pareto Optimal when there is no other allocation x' where every agent gets at least as much utility as in x, and at least one agent gets higher utility. In the restricted valuations case, every item adds valuation either 0 or v_j to some agent's utility. Thus, the sum of valuations of all agents in any allocation is at most $\sum_j v_j$. SMatch can easily be modified so that it allocates every item to some agent who has non zero valuation for it. Then, the sum of valuations of all agents in the allocation returned by SMatch is $\sum_j v_j$. No other allocation can give an agent strictly higher utility without decreasing another agent's utility. Hence, x^* is a Pareto Optimal allocation. QED.

Remark 3.3.2. *We note here that Theorem 3.3.4 holds for general additive valuations also. However for the general case, the* PO *property does not always hold. Consider for example the case where we have two agents,* A *and* B *and four items,* $\{g_1, g_2, g_3, g_4\}$. *Agent* A *values the items at* $\{2+\epsilon, 2, \epsilon, \epsilon\}$ *and agent* B *values them at* $\{1, 1, 1, 1\}$. SMatch *allocates items* g_1, g_3 *to agent* A *and items* g_2, g_4 *to agent* B. *However, we can swap items* g_2 *and* g_3 *to get an allocation that pareto dominates the allocation output by the algorithm.*

3.4 ADDITIVE VALUATIONS

In this section, we present SMatch, described in Algorithm 3, for the asymmetric additive NSW problem, and prove the following approximation result.

Theorem 3.4.1. *If* $\mathbf{x}$ *denotes the output allocation of* SMatch, $\mathsf{NSW}(\mathbf{x}) \geq \frac{1}{2\mathsf{n}}\mathsf{OPT}$.

3.4.1 Algorithm

SMatch works in a single pass. For every agent, the algorithm first computes the value of $m - 2n$ least valued items and stores this in u_i. SMatch then defines a weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{M}, \mathcal{W})$ similarly as in the submodular case, but here the edge weights are defined as $w(i,j) = \eta_i \log\left(v_i(j) + \frac{u_i}{n}\right)$, and allocates one item to each agent along the edges of a maximum weight matching of Γ. It then starts allocating items via repeated matchings. Until all items are allocated, SMatch iteratively defines graphs $\Gamma(\mathcal{A}, \mathcal{M}^{rem}, \mathcal{W})$ with $\mathcal{M}^{rem}$ denoting the set of unallocated items and edge weights defined as $w(i,j) = \eta_i \log\left(v_i + v_i(j)\right)$, where v_i is the valuation of agent i for items that are allocated to her. SMatch then allocates at most one item to each agent according to a maximum weight matching of Γ.

Algorithm 3: SMatch for the Asymmetric Additive NSW problem

Input : A set $\mathcal{A}$ of n agents with weights $\eta_i,\ \forall i \in \mathcal{A}$, a set $\mathcal{M}$ of m indivisible items, and additive valuations $v_i : 2^{\mathcal{M}} \to \mathbb{R}_+$, where $v_i(\mathcal{S})$ is the valuation of agent $i \in \mathcal{A}$ for a set of items $\mathcal{S} \subseteq \mathcal{M}$.

Output: An allocation that approximately optimizes the NSW.

$\mathbf{x}_i \leftarrow \emptyset, u_i \leftarrow v_i(\mathcal{M}_{i,[2n+1:m]}) \quad \forall i \in [n]$ // $\mathcal{M}_{i,[a:b]}$ defined in Section 3.4.2

Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{M}, \mathcal{W})$ with weights $\mathcal{W} = \{w(i,j) \mid w(i,j) = \eta_i \log\left(v_i(j) + \frac{u_i}{n}\right), \forall i \in \mathcal{A}, j \in \mathcal{M}\}$

Compute a maximum weight matching $\mathcal{M}$ for Γ

$\mathbf{x}_i \leftarrow \mathbf{x}_i \cup \{j \mid (i,j) \in \mathcal{M}\},\ \forall i \in \mathcal{A}$ // allocate items according to $\mathcal{M}$

$\mathcal{M}^{rem} \leftarrow \mathcal{M} \backslash \{j \mid (i,j) \in \mathcal{M}\}$ // update set of unallocated items

while $\mathcal{M}^{rem} \neq \emptyset$ **do**

  Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{M}^{rem}, \mathcal{W})$ with weights $\mathcal{W} = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(j) + v_i(\mathbf{x}_i)), i \in \mathcal{A}, j \in \mathcal{M}^{rem}\}$

  Compute a maximum weight matching $\mathcal{M}$ for Γ

  $\mathbf{x}_i \leftarrow \mathbf{x}_i \cup \{j \mid (i,j) \in \mathcal{M}\},\ \forall i \in \mathcal{A}$ // allocate items according to $\mathcal{M}$

  $\mathcal{M}^{rem} \leftarrow \mathcal{M}^{rem} \backslash \{j \mid (i,j) \in \mathcal{M}\}$ // remove allocated items

end

Return $\mathbf{x}$

3.4.2 Notation

In the following discussion, we use $\mathbf{x}_i = \{h_i^1, \ldots, h_i^{\tau_i}\}$ to denote the set of items received by agent i in SMatch. We use $\mathbf{x}_i^* = \{g_i^1, \ldots, g_i^{\tau_i^*}\}$ to denote the set of items in i's optimal bundle. τ_i and τ_i^* , are the number of items in $\mathbf{x}_i$ and $\mathbf{x}_i^*$ respectively. Then for every i, all items in $\mathbf{x}_i$ and $\mathcal{M}$ are ranked according to the decreasing utilities as per v_i. We use the shorthand $[n]$ to denote the set $\{1, 2, \ldots, n\}$. $\mathcal{M}_{i,[a:b]}$ denotes the items ranked from a to b according to agent i in $\mathcal{M}$, and $\mathbf{x}_{i,1:t}$ is the total allocation to agent i from the first t matching iterations. We also use $\mathcal{M}_{i,k}$ to denote the k^{th} ranked item of agent i from the entire set of items. For all i, we define u_i as the minimum value for the remaining set of items upon removing at most $2n$ items from $\mathcal{M}$, i.e., $u_i = \min_{\mathcal{S} \subseteq \mathcal{M}, |\mathcal{S}| \leq 2n} v_i(\mathcal{M} \setminus \mathcal{S}) = \mathcal{M}_{i,[2n+1,m]}$.[3]

3.4.3 Analysis

To establish the guarantee of Theorem 3.4.1, we first prove a couple of lemmas.

Lemma 3.4.1. $v_i(h_i^t) \geq v_i(\mathcal{M}_{i,tn})$.

[3] As the valuation functions are monotone, the minimum value will be obtained by removing exactly $2n$ items. The less than accounts for the case when the number of items in $\mathcal{M}$ is fewer than $2n$.

Proof. Since every iteration of SMatch allocates at most n items, at the start of iteration t at most $(t-1)n$ items are allocated. Thus at least n items from $\mathcal{M}$ ranked between 1 to tn by agent i are still unallocated. In the t^{th} iteration the agent will thus get an item with value at least the value of $\mathcal{M}_{i,tn}$ and the lemma follows. QED.

Lemma 3.4.2. $v_i(h_i^2, \ldots, h_i^{\tau_i}) \geq \frac{u_i}{n}$.

Proof. Using Lemma 3.4.1 and since $v_i(\mathcal{M}_{i,tn}) \geq v_i(\mathcal{M}_{i,tn+k}),\ \forall k \in [n-1]$

$$v_i(h_i^t) \geq \frac{1}{n} v_i(\mathcal{M}_{i,[tn:(t+1)n-1]}) \ .$$

Thus,

$$v_i(h_i^2, \ldots, h_i^{\tau_i}) = \sum_{t=2}^{\tau_i} v_i(h_i^t) \geq \frac{1}{n} \sum_{t=2}^{\tau_i} (v_i(\mathcal{M}_{i,[tn:(t+1)n-1]})$$

As at most n items are allocated in every iteration, agent i receives items for at least $\lfloor \frac{m}{n} \rfloor$ iterations.[4] This implies that $(\tau_i + 1)n \geq m$ and hence,

$$\begin{aligned} v_i(h_i^2, \ldots, h_i^{\tau_i}) &\geq \frac{1}{n}\Big(v_i(\mathcal{M}_{i,[2n:m-1]})\Big) \\ &\geq \frac{1}{n}\Big(v_i(\mathcal{M}_{i,[2n+1:m]})\Big) = \frac{1}{n} u_i. \end{aligned}$$

The second inequality follows as $v_i(\mathcal{M}_{i,2n}) \geq v_i(\mathcal{M}_{i,m})$. QED.

We now prove our main theorem.

Proof of Theorem 3.4.1.

$$\begin{aligned} \mathsf{NSW}(\mathbf{x}) &= \prod_{i=1}^{n} \left(v_i\left(h_i^1, \ldots, h_i^{\tau_i}\right)^{\eta_i}\right)^{\frac{1}{\sum_{i=1}^{n} \eta_i}} \\ &= \prod_{i=1}^{n} \left(\left(v_i(h_i^1) + v_i(h_i^2 \ldots, h_i^{\tau_i})\right)^{\eta_i}\right)^{\frac{1}{\sum_{i=1}^{n} \eta_i}} \\ &\geq \prod_{i=1}^{n} \left(\left(v_i(h_i^1) + \frac{u_i}{n}\right)^{\eta_i}\right)^{\frac{1}{\sum_{i=1}^{n} \eta_i}}, \end{aligned}$$

where the last inequality follows from Lemma 3.4.2. During the allocation of the first item h_i^1, items g_i^1 of all agents are available. Thus, allocating each agent her own g_i^1 is a feasible first

[4]Here we assume that the agents have non-zero valuation for every item. If it does not, the other case is also straightforward and the lemma continues to hold.

matching and we get

$$\mathsf{NSW}(\mathbf{x}) \geq \prod_{i=1}^{n} \left(\left(v_i(g_i^1) + \frac{u_i}{n} \right)^{\eta_i} \right)^{\frac{1}{\sum_{i=1}^{n} \eta_i}}.$$

Now, $u_i = \min_{\mathcal{S} \subseteq \mathcal{M}, |\mathcal{S}| \leq 2n} v_i(\mathcal{M} \setminus \mathcal{S})$. Suppose we define, $\mathcal{S}_i^* = \arg\min_{|\mathcal{S}| \leq 2n, \mathcal{S} \subseteq \mathbf{x}_i^*} v_i(\mathbf{x}_i^* \setminus \mathcal{S})$, then $v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i^*) \leq u_i$. To see this, let $\mathcal{S}_i = \arg\min_{\mathcal{S} \subseteq \mathcal{M}, |\mathcal{S}| \leq 2n} v_i(\mathcal{M} \setminus \mathcal{S})$. Now, $u_i = v_i(\mathcal{M} \setminus \mathcal{S}_i) \geq v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i) \geq v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i^*)$. Thus,

$$\begin{aligned}
\mathsf{NSW}(\mathbf{x}) &\geq \prod_{i=1}^{n} \left(\left(\frac{1}{2n} v_i(\mathcal{S}_i^*) + \frac{1}{n} v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i^*) \right)^{\eta_i} \right)^{\frac{1}{\sum_i \eta_i}} \\
&\geq \frac{1}{2n} \prod_{i=1}^{n} \left(v_i(\mathbf{x}_i^*)^{\eta_i} \right)^{\frac{1}{\sum_{i=1}^{n} \eta_i}} \\
&= \frac{1}{2n} \mathsf{OPT}. \qquad \text{QED.}
\end{aligned}$$

Remark 3.4.1. SMatch *easily extends to give an $O(n)$ approximation for the case of budget-additive valuations. The changes required in the algorithm are $u_i = min(c_i, \mathcal{M}_{1,2n+1:m})$ where c_i is the utility cap for agent i. Also, the edge weights in the bipartite graphs will use marginal utility (as they had in submodular valuations case). Lemma 3.4.2 and the subsequent proof can be extended easily for budget-additive by combining ideas from Lemma 3.1.2 and Proof of Theorem 3.1.1.*

3.5 TIGHTNESS OF THE ANALYSIS

3.5.1 Subadditive Valuations

The matching approach does not extend to agents with subadditive valuation functions. Here the valuation functions satisfy the subadditivity property:

$$v(\mathcal{S}_1 \cup \mathcal{S}_2) \leq v(\mathcal{S}_1) + v(\mathcal{S}_2),$$

for any subsets $\mathcal{S}_1, \mathcal{S}_2$ of the set of items $\mathcal{M}$.

A counter example that exhibits the shortcomings of the approach is as follows. Consider an instance with 2 agents and m items. Assume m is even. Denote the set of items by $\mathcal{M} = \{g_1, g_2, \ldots, g_m\}$. Let $\mathcal{M}_1 = \{g_1, g_2, \ldots, g_{m/2}\}$ and $\mathcal{M}_2 = \{g_{m/2+1}, \ldots, g_m\}$. The valuation func-

tion for agent $i \in \{1, 2\}$ is as follows.

$$v_i(\mathcal{S}) = v_i(\mathcal{S}_1 \cup \mathcal{S}_2) = \max\{M, |\mathcal{S}_i| \cdot M\} \quad \forall \mathcal{S} \subseteq \mathcal{M},\, \mathcal{S}_1 \subseteq \mathcal{M}_1,\, \mathcal{S}_2 \subseteq \mathcal{M}_2.$$

Note that these valuation functions are subadditive.

The allocation that maximizes the NSW allocates $\mathcal{M}_1$ to agent 1 and $\mathcal{M}_2$ to agent 2. The optimal NSW is $mM/2$.

Now, RepReMatch may proceed in the following way. Since the marginal utility of each item over ϕ is M, the algorithm can pick any of the items for either of the agents. Suppose the algorithm gives $g_{m/2+1}$ to agent 1 and g_1 to agent 2. In the next iteration, for agent 1 (2) the marginal utility of any item over $g_{m/2+1}$ (g_1) is 0. Thus, again the algorithm is at liberty to allocate any item to either of the agents. Now again the algorithm gives exactly opposite allocation as compared to the optimal allocation and gives agent 1 item $g_{m/2+2}$ and gives agent 2 item g_2. For each iteration this process repeats and ultimately the bundles allocated by algorithm are exactly opposite of the bundles allocated by optimal. The re-matching step first releases $\log n + 1$, or 2 items from both agent allocations, and re-matches them. This may not change the allocations as both agents have already received their best item. The NSW of the algorithm's allocation is $(M^2)^{1/2} = M$, giving an approximation ratio of $\Omega(m)$ with the optimal NSW. Even if we increase the number of agents, the factor cannot be made independent of m, the number of items.

The problem in the subadditive case is the myopic nature of each iteration in RepReMatch. In each iteration the algorithm only sees one step ahead. At any of the iterations, had the algorithm been allowed to pick and allocate multiple items instead of 1, it would have been able to select a subset of items from it's correct optimal bundle.

This problem does not arise in the additive case because the valuation of an item here is independent of other items. Submodular valuations allow a minimum marginal utility over an agent's current allocation for items allocated in future iterations, hence this issue does not arise there too.

3.5.2 XOS Valuations

The following example tells us that RepReMatch does not extend to XOS valuations either. XOS is a class of valuation functions that falls between subadditive and submodular valuation functions, defined as follows. A set of additive valuation functions, say $\{\ell_1, \ldots, \ell_k\}$, is given, and the XOS valuation of a set of items $\mathcal{S}$ is the maximum valuation of this set according to any of these additive valuations. i.e., $v(\mathcal{S}) = \max_{i \in [k]}\{\ell_i(\mathcal{S})\}$.

To see why the algorithm does not extend to this class of functions, consider the following counter example. We have $n = 2$ agents and $m = 2k$ items, for some $k > 3$. Each agent $i \in \{1, 2\}$

has 2 valuation functions ℓ_1^i, ℓ_2^i. The following two tables pictorially depict these valuations. Each entry $(\ell_h^i, g_j),\ h \in [2], i \in [2], j \in [2k]$ denotes agent i's valuation according to function ℓ_h^i for item g_j.
For agent 1:

	g_1	g_2	$\dots$	g_k	g_{k+1}	g_{k+2}	g_{k+3}	g_{k+4}	$\dots$	g_{2k}
ℓ_1^1	M	M	$\dots$	M	0	0	0	0	$\dots$	0
ℓ_2^1	0	0	$\dots$	0	M + ϵ	M + ϵ	M + ϵ	ϵ	$\dots$	ϵ

For agent 2:

	g_1	g_2	g_3	g_4	$\dots$	g_k	g_{k+1}	g_{k+2}	$\dots$	g_{2k}
ℓ_1^2	0	0	0	0	$\dots$	0	M	M	$\dots$	M
ℓ_2^2	M+ϵ	M+ϵ	M+ϵ	ϵ	$\dots$	ϵ	0	0	$\dots$	0

Here M is any large value, and $\epsilon > 0$ is negligible.

The allocation optimizing NSW clearly allocates the first k items to agent 1 and the next k items to agent 2, resulting in the NSW value Mk. RepReMatch on the other hand allocates items g_1, g_2 to agent 2 and items g_{k+1}, g_{k+2} to agent 1 in Phase 1. In Phase 2, it gives g_3 to agent 2 and g_{k+3} to agent 1. After this, for all other iterations of Phase 2, items $g_j, j \leq k$ have zero marginal utility for agent 1 and items $g_j, j \geq (k+1)$ have zero marginal utility for agent 2. Thus, RepReMatch allocates items $g_3 \dots g_k$ to agent 2 and items $g_{k+3}, \dots, g_{2k}$ to agent 1 in Phase 2. Phase 3 reallocates items of Phase $1 - g_1, g_2, g_{k+1}, g_{k+2}$ allocating g_{k+1}, g_{k+2} to agent 1 and g_1, g_2 to agent 2. Thus the NSW of the allocation given by RepReMatch is $(3M + k \cdot \epsilon)$. Hence, the approximation ratio of RepReMatch cannot be better than $(MK)/(3M + k\epsilon)$ or $\Omega(k) = \Omega(m)$ when the valuation functions are XOS.

3.5.3 Asymmetric Additive NSW

We describe an example to prove the analysis of this case is tight. Consider an NSW instance with n agents, referred by $\{1, 2 \dots, n\}$ and m sets of n^2 items, referred by $\mathcal{M} = \{\mathcal{M}_i \mid i \in [m]\}$, where every $\mathcal{M}_i = \{g_{i,1} \dots, g_{i,n^2}\}\}$. The first agent has weight W, while the remaining agents have weight 1. The valuation function of agent 1 is as follows.

$$v_1(g_{i,j}) = \begin{cases} M & j \in [n], i \in [m] \\ 0 & otherwise. \end{cases}$$

The remaining agents have valuations for items as follows.

$$\forall k \in [n], k \neq 1 :\ v_k(g_{i,j}) = \begin{cases} M + \epsilon & j \in [n], i \in [m], \epsilon > 0 \\ M + \bar{\epsilon} & (k-1)n + 1 \leq j \leq kn, i \in [m], \epsilon > \bar{\epsilon} > 0 \\ 0 & otherwise. \end{cases}$$

It is easy to verify that the optimal allocation that maximizes NSW gives all items $g_{i,j}$ for i between $(k-1)n+1$ and kn to agent k. That is, the k^{th} agent receives the k^{th} set of n items from each of the m sets of n^2 items. SMatch on the other hand allocates items as follows. For the first graph, it computes u_i as $M(m-2)n$ for the first agent, and $(M+\epsilon)(m-2)n + (M+\epsilon')(mn)$ for a small $\epsilon' > 0$, for the rest. The first max weight matching then allocates to each agent one item from agent 1's optimal bundle. The following iterations also err as follows. Until the first n items from every set are allocated, irrespective of the agent weights, every agent receives one item from this set if available. In the remaining matchings, agent 1 does not receive any item, and the other agents get all items in their optimal bundles. The ratio of the NSW products of the optimal allocation and the algorithm's allocation is as follows.

$$\begin{aligned} \frac{\mathsf{NSW}(\mathbf{x})}{\mathsf{OPT}} &\leq \left(\frac{(mM)^W (mn \cdot (M+\bar{\epsilon}) + m(M+\epsilon))^{n-1}}{(mn \cdot M)^W (mn \cdot (M+\bar{\epsilon}))^{n-1}} \right)^{1/(W+(n-1))} \\ &\leq \left(\frac{(mM)^W (2mn \cdot M)^{n-1}}{(mn \cdot M)^W (mn \cdot M)^{n-1}} \right)^{1/(W+(n-1))} \\ &\leq 2 \left(\frac{1}{n} \right)^{W/(W+(n-1))}. \end{aligned}$$

With increasing W, asymptotically this ratio approaches $2/n$.

Remark 3.5.1. *It is natural to ask if the asymmetric* NSW *problem is harder than the symmetric problem. As* SMatch *is the first non-trivial algorithm for the asymmetric problem, we would like to find if it gives a better approximation factor when applied to a symmetric agents instance. However, after considerable effort, we could not resolve this question definitively. Like the above example, we could not find an example of a symmetric instance for which our analysis was tight. Our conjecture is that* SMatch *gives a better factor for the symmetric problem, and that the symmetric case itself is easier than the asymmetric* NSW *case.*

CHAPTER 4: MAXIMIN SHARE WITH A MIXED MANNA

In this chapter, we explore the maxmin share fairness notion. To recall, the maximin share (MMS) value of an agent is defined as follows. If $\Pi_n(\mathcal{M})$ represents all possible partitions $(A_1, \ldots, A_n)$ of $\mathcal{M}$ into n bundles, and v_i is her valuation function, then

$$\mathsf{MMS}_i(\mathcal{M}) = \max_{(A_1,\ldots,A_n)\in\Pi_n(\mathcal{M})} \min_{k\in[n]} v_i(A_k) \ . \tag{4.1}$$

An MMS *allocation* is one where every agent gets at least her MMS value. We initiate the study of finding MMS + PO allocations for a mixed manna.

We first show that, for any fixed $\alpha \in (0, 1]$, an α-MMS allocation may not always exist (see Section 4.3). This rules out efficient computation for any fixed α, and naturally raises the following problem.

Design an efficient algorithm to find an α-MMS + PO *allocation for the best possible* α*, i.e., the maximum* $\alpha \in (0, 1]$ *for which it exists.*

This *exact* problem is intractable: In the case of identical agents, an $(\alpha = 1)$-MMS allocation exists by definition. However, finding one is known to be NP-hard even for a goods manna. Also, checking if a given instance admits an MMS allocation is known to be in $\mathsf{NP}^{\mathsf{NP}}$, but not known to be in NP ([46]). On the positive side, a polynomial-time approximation scheme (PTAS) is known for this case due to [45]; given a *constant* $\epsilon \in (0, 1]$, the algorithm finds a $(1 - \epsilon)$-MMS allocation in polynomial time. No such result is known when the agents are not identical. Guaranteeing PO in addition adds to the complexity, since even checking if a given allocation is PO is coNP-hard even with two identical agents ([75]). In light of these results, we ask,

Question. *Can we design a* PTAS*, namely an efficient algorithm to find an* $(\alpha - \epsilon)$-MMS + γ-PO *allocation, given* $\epsilon, \gamma > 0$*, for the best possible* α*?*

Our Contribution. We show the following dichotomy result: We derive two conditions and show that the problem is tractable under these conditions, while dropping either renders the problem intractable. The two conditions are: (i) number of agents n is a constant, and (ii) for every agent i, her total (absolute) value for all the items ($|v_i(\mathcal{M})|$) is significantly greater than the minimum of her total value of goods (v_i^+) and her total (absolute) value for chores (v_i^-), i.e., for a constant $\tau > 0$, $|v_i(\mathcal{M})| \geq \tau \cdot \min\{v_i^+, v_i^-\}$.

In particular, first, for instances satisfying (i) and (ii), we design a PTAS (as asked in the above question). Second, we show that if either condition is not satisfied, then finding an α-MMS allocation for *any* $\alpha \in (0, 1]$ is NP-hard, even with *identical agents* where a solution exists for $\alpha = 1$. This hardness is striking because it shows inapproximability within *any* non-trivial factor when either (i) or (ii) is not satisfied. This also indicates that the two conditions are unavoidable.

Our algorithm, in principle, gives a little more than a PTAS. It runs in time $2^{O(1/\min\{\epsilon^2,\gamma^2\})}poly(\mathrm{m})$ for given $\epsilon,\ \gamma$, thus gives polynomial run-time for ϵ, γ as small as $O(1/\sqrt{\log m})$, where $m = |\mathcal{M}|$.

α-MMS+PO for goods (chores) manna. As a corollary, we obtain a PTAS for finding α-MMS+PO allocations of a goods manna and a chores manna when the number of agents is a constant. This improves the previous results for these settings in two aspects: (i) provides the best possible approximation factor; factors better than the general case known for good manna are $4/5$ for $n = 4$ by [50], $8/9$ for $n = 3$ by [56], and 1 for $n = 2$ by [46], and (ii) provides an additional (approximate) PO guarantee.

Challenges. The key challenge in solving this question is handling items of high value to any agent. In the goods or chores mannas, these items can be greedily assigned, for example as singleton bundles. But in a mixed manna, *high valued* goods (chores) may have to be bundled with specific sets of chores (goods) or low valued items to form lesser valued bundles. Secondly, the MMS values of the agents, and the α for which α-MMS allocation exist, both are not known. In fact, computing the exact MMS values is NP-hard (even with a goods manna).

PTAS to find MMS values. As the first key step for our main algorithm, we design a PTAS that returns $(1-\epsilon)$ approximate MMS values of agents, which may be of independent interest.

A new technique to prove PO. Since certifying a PO allocation is a coNP-hard problem ([75]), known works (e.g., [30, 76, 77]) maintain a PO allocation with market equilibrium as a certificate . We develop a novel approach to ensure PO with α-MMS through LP rounding. The LP itself is intuitive, however the rounding is involved. It makes use of *envy-graph* and properties of the MMS in a novel way. This approach may be of independent interest.

Organization. Section 4.1 gives a formal definition of the problem and notations. Section 4.2 discusses the main result of PTAS for the α-MMS + PO problem with the best possible α. The formal discussion of the PTAS for computing MMS values for the case when MMS ≥ 0 is in Section 4.4 and for the MMS < 0 case is in Section 4.5. Section 4.3 discusses the non-existence of α-MMS allocation for any $\alpha \in (0, 1]$ and Section 4.6 discusses the NP-hardness results.

4.1 PROBLEM DEFINITION AND NOTATIONS

Notations. We use $[k]$ to denote the set $\{1, 2 \cdots, k\}$. For $c \in \mathbb{R}$, c^+ denotes $\max\{c, 0\}$.

We consider the problem of allocating a set $\mathcal{M}$ of m indivisible items among a set $\mathcal{N}$ of n agents in a *fair* and *efficient* manner, with the fairness notion of *maximin share* (MMS) and the efficiency notion of *Pareto-optimality* (PO). Each agent $i \in \mathcal{N}$ has an additive valuation function $v_i : 2^{\mathcal{M}} \to \mathbb{R}$ over sets of items. For a set $S \subseteq \mathcal{M}$, her value is $v_i(S) = \sum_{j \in S} v_{ij}$. Agents are called *identical* if their v_is are the same function; in this case, the valuation function is denoted by v.

The set of items valued non-negatively (negatively) by an agent i are called her *Goods* (*Chores*), and denoted by $\mathcal{M}_i^+ = \{j \mid v_{ij} \geq 0\}$ ($\mathcal{M}_i^- = \{j \mid v_{ij} < 0\}$). The sets of all the goods and all the chores of the instance are defined as respectively $\mathcal{M}^+ := \cup_i \mathcal{M}_i^+$, and $\mathcal{M}^- := \mathcal{M} \backslash \mathcal{M}^+$. We refer to an item j as a *good* if $v_{ij} \geq 0$ for *some* agent and as a *chore* if $v_{ij} < 0$ for all agents.

MMS values and allocation. Let $A^\pi = \{A_1, A_2, \cdots, A_n\}$ denote a partition of all the items among the n agents, referred as an *allocation*, i.e., $A_i \cap A_{i'} = \emptyset$ for all distinct i, i' in $\mathcal{N}$, and $\cup_i A_i = \mathcal{M}$. And let $\Pi_n(\mathcal{M})$ be the set of all possible allocations of $\mathcal{M}$ among n agents. The maximin share (MMS) value of an agent i is defined as

$$\mathsf{MMS}_i^n(\mathcal{M}) = \max_{(A_1,\ldots,A_n)\in\Pi_n(\mathcal{M})} \min_{k\in[n]} v_i(A_k).$$

We refer to $\mathsf{MMS}_i^n(\mathcal{M})$ by MMS_i when the qualifiers n and $\mathcal{M}$ are clear, and by MMS when agents are identical. Note that MMS_i can be negative too.

Definition 4.1.1 (α-MMS allocation)**.** *A^π is called an α-MMS allocation for an $\alpha \in (0, 1]$, if for each agent $i \in \mathcal{N}$ we have $v_i(A_i) \geq \alpha\mathsf{MMS}_i$ if $\mathsf{MMS}_i \geq 0$, $v_i(A_i) \geq (1/\alpha)\mathsf{MMS}_i$, if $\mathsf{MMS}_i < 0$. Equivalently, $v_i(A_i) \geq \min\{\alpha\mathsf{MMS}_i, (1/\alpha)\mathsf{MMS}_i\}$. When $\alpha \leq 0$, for simplicity, we define any allocation as α-MMS.*

γ-Pareto optimal (γ-PO) and γ-Pareto dominating allocations. An allocation A^π is said to be γ-PO if there does not exist any $B^\pi \in \Pi_n(\mathcal{M})$, called an allocation γ-Pareto dominating A^π, such that $\forall i \in \mathcal{N},\ v_i(B_i) \geq (1+\gamma)v_i(A_i)$ if $v_i(A_i) \geq 0$, and $v_i(B_i) \geq \frac{1}{(1+\gamma)}v_i(A_i)$ if $v_i(A_i) < 0$, and for at least one i the inequality is strict.

An allocation is called PO if it is 0-PO. It is easy to see that if there exists an α-MMS allocation for a given instance then there is one that is both α-MMS and PO (and thereby also γ-PO). This is because if an allocation B^π Pareto dominates an α-MMS allocation A^π, then B^π is also α-MMS.

Since the problem of finding α-MMS allocation is NP-hard for any $\alpha \in (0, 1]$, we design a PTAS to compute an α-MMS + PO allocation for a sub-class of instances. To characterize this sub-class, we will need the following definition.

For each agent $i \in \mathcal{N}$, define $v_i^+ = \sum_{j\in\mathcal{M}_i^+} v_{ij}$ and $v_i^- = \sum_{j\in\mathcal{M}_i^-} |v_{ij}|$. (4.2)

Definition 4.1.2 (α-MMS + PO Problem)**.** *Given an instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$ and $\alpha \in (0, 1]$ where,*

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

either find an allocation $A^\pi \in \Pi_n(\mathcal{M})$ that is both α-MMS and PO, also called an α-MMS + PO allocation, or correctly report that such an allocation does not exist for the given instance.

The above problem without the PO guarantee is called the α-MMS problem. Unlike the goods manna or the chores manna, for the mixed manna an α-MMS allocation may not exist for *any* $\alpha > 0$, as shown in Electronic Companion Section 4.3. Therefore, for a mixed manna, we can only hope to find an α-MMS allocation for the maximum possible α value for the given instance, formally defined below.

Definition 4.1.3 (OPT-α-MMS + PO Problem.)**.** *Given an instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$, find an allocation which is α-MMS + PO for an $\alpha \in (0, 1]$ such that there is no α'-MMS allocation for any $\alpha' > \alpha$.*

Note that, given an algorithm for the α-MMS+PO problem it is easy to solve OPT-α-MMS+PO by doing a binary search on the value of α. We design a PTAS for the former in Section 4.2 and thereby solve the latter efficiently up to a small error. By PTAS we mean, given constants $\epsilon, \gamma > 0$, in polynomial-time it either returns an $(\alpha - \epsilon)^+$-MMS + γ-PO allocation, or correctly reports that no α-MMS allocation exists.

The following observations will be useful in what follows.

Lemma 4.1.1. *$v_i(\mathcal{M}) \geq 0$ iff $\mathsf{MMS}_i \geq 0$.*

Proof. If the sum of valuations of all items $v_i(\mathcal{M})$ is negative, there can be no allocation where every bundle has non-negative valuation. Hence, MMS_i is negative. If the sum of valuations is positive, then adding all items to one bundle and no item in other bundles makes the least-valued bundle have zero value. Thus, in this case, $\mathsf{MMS}_i \geq 0$. QED.

Lemma 4.1.2. *$\mathsf{MMS}_i \leq v_i(\mathcal{M})/|\mathcal{N}|$ for all $i \in \mathcal{N}$.*

Proof. If $\mathsf{MMS}_i > v_i(\mathcal{M})/n$, it implies that there exists a partition of items in $\Pi_n(\mathcal{M})$ where all bundles have value greater than $v_i(\mathcal{M})/n$. Therefore, $v_i(\mathcal{M}) \geq n.\mathsf{MMS}_i > n.\frac{v_i(\mathcal{M})}{n} = v_i(\mathcal{M})$, which is a contradiction. QED.

Lemma 4.1.3. *[Scale Invariance] α-MMS + PO allocations for the instances $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$ and $(\mathcal{N}, \mathcal{M}, (v'_i)_{i\in\mathcal{N}})$ are the same when for all i, j, $v'_{ij} = c_i \cdot v_{ij}$ for some constants $c_i > 0$.*

Proof. For any agent i, the value of any bundle of items $\mathcal{S}$ according to the two valuation function are related as $v'_i(\mathcal{S}) = c_i \cdot v_i(\mathcal{S})$. Thus, by definition of MMS, her MMS values according to the two valuation functions are also related as $\mathsf{MMS}'_i = c_i \cdot \mathsf{MMS}_i$, where MMS'_i is agent i's MMS value according to v'_i.

This implies that a set of items has α-MMS value for i according to $(v_i)_{i\in\mathcal{N}}$ if and only if it has α-MMS value according to $(v'_i)_{i\in\mathcal{N}}$. Hence all α-MMS allocations according to valuations $(v_i)_{i\in\mathcal{N}}$ are also α-MMS according to $(v'_i)_{i\in\mathcal{N}}$ and vice versa.

The α-MMS + PO allocation according to $(v_i)_{i\in\mathcal{N}}$, say $\mathcal{A}$, is α-MMS and must also be PO according to $(v'_i)_{i\in\mathcal{N}}$, as otherwise the Pareto dominating allocation will Pareto dominate $\mathcal{A}$ according to $(v_i)_{i\in\mathcal{N}}$ too. QED.

4.2 PTAS FOR α-MMS + PO WITH NON-IDENTICAL AGENTS

In this section, we present our main result, namely a PTAS for the OPT-α-MMS + PO problem.

For OPT-α-MMS+PO the crucial step is to get a PTAS for the α-MMS+PO problem, discussed in Electronic Companion Sections 4.4 and 4.5. Recall that the definition of the latter, namely Definition 4.1.2, assumes two conditions on the input instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$: (i) number of agents is a constant, and (ii) for each $i \in \mathcal{N}$, $|v_i(\mathcal{M})| \geq \tau \cdot \min\{v_i^+, v_i^-\}$, where $\tau > 0$ is a constant. Let us first briefly discuss why both of these conditions are unavoidable.

Hardness of approximation. In Electronic Companion Section 4.6, we show the following theorem by proving that if either condition is dropped then the problem is intractable for *any* $\alpha \in (0, 1]$, even when exact MMS allocation exists.

Theorem 4.2.1. *For any instance $(n, \mathcal{M}, v)$ with identical agents and $v(\mathcal{M}) > 0$ such that exactly one of the following two holds: (a) either $n = 2$ or (b) $|v(\mathcal{M})| \geq \tau \cdot \min\{v(\mathcal{M}^+), |v(\mathcal{M}^-)|\}$ for a constant τ, finding an α-MMS allocation of $(n, \mathcal{M}, v)$ for any $\alpha \in (0, 1]$ is NP-hard.*

To prove the above theorem, we design two reductions from a well-known NP-hard problem PARTITION to the problem of finding an α-MMS allocation of an instance $(n, \mathcal{M}, v)$ for any $\alpha \in (0, 1]$,. The tricky part in these reductions is to guarantee that an α-MMS allocation for *any* $\alpha > 0$ maps to a solution of PARTITION.

Computing the MMS values. The first step in our PTAS is to compute the MMS values of the agents, which is equivalent to finding an MMS allocation with identical agents. The above hardness result rules out even approximating the MMS values within any non-trivial factor in polynomial time if either condition is not satisfied. For the instances satisfying both, in Electronic Companion Sections 4.4 and 4.5 we design an efficient algorithm to compute the MMS values up to a small multiplicative error. We need to tackle the cases with MMS ≥ 0 and MMS < 0 separately; note that the sign of MMS can be easily determined using Lemma 4.1.1. Formally, we show the following (see Electronic Companion Sections 4.4 and 4.5).

Theorem 4.2.2. *Given an instance $(n, \mathcal{M}, v)$ and a constant $\epsilon > 0$, if (i) n is a constant and (ii) for each $i \in [n]$, $|v_i(\mathcal{M})| \geq \tau \cdot \min\{v_i^+, v_i^-\}$, where $\tau > 0$ is a constant. Then, there is a* PTAS *to compute a $(1-\epsilon)$-*MMS *allocation.*

Our PTAS for the α-MMS$+$PO problem takes as input the instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$, a parameter $\alpha \in (0, 1]$, and constants $\epsilon, \gamma > 0$, and it either finds an allocation that is $(\alpha-\epsilon)^+$-MMS$+\gamma$-PO allocation, or correctly reports that an α-MMS allocation does not exist; the latter may very well be the case for *any* $\alpha \in (0, 1]$ as shown in Electronic Companion Section 4.3.

Pre-processing. First, note that the problem is non-trivial only if $\alpha > \epsilon$, otherwise since $(\alpha - \epsilon)^+ = 0$, thus an allocation that gives every item to the agent with the highest value for it is $(\alpha-\epsilon)^+$-MMS $+$ PO, and returned. Therefore, now on we assume that $\alpha > \epsilon$.

Next we re-define ϵ as $\min\{\epsilon, \frac{\gamma\alpha}{(1+\gamma)}\}$. This is done for technical reasons to ensure that the final allocation is also γ-PO. It does not harm the MMS guarantee, as an $(\alpha-\epsilon)^+$-MMS allocation with a smaller ϵ is also an $(\alpha-\epsilon)^+$-MMS allocation with respect to the given ϵ. Note that when α and γ are constants, so is the new value of ϵ. Finally, we assume there are no agents with $v(\mathcal{M}) = 0$. Note that because of condition 2 of the problem, when $v(\mathcal{M}) = 0$ then the value of every item for this agent is 0. Also note that their MMS $= 0$. Thus, we can allocate all the chores arbitrarily among agents with $v(\mathcal{M}) = 0$, and remove them. It is easy to see that the MMS value of the remaining agents can only improve, and all α-MMS allocations are retained, by the removal of all the chores and a subset of agents. The problem then reduces to a goods manna case with no agents with $v(\mathcal{M}) = 0$, which is solved as a special case of the PTAS we will describe.

Due to the pre-processing step, now on we assume that $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$, the given fair division instance, satisfies $v_i(\mathcal{M}) \neq 0$ for every agent $i \in \mathcal{N}$. We first scale the valuations so that $|v_i(\mathcal{M})| = n$ since the problem is scale free by Lemma 4.1.3. Without loss of generality, we assume that the given constants $\alpha, \epsilon, \gamma > 0$ are such that $\alpha > \epsilon$, and $\epsilon \leq \frac{\gamma\alpha}{(1+\gamma)}$. The algorithm first applies the relevant PTAS from Electronic Companion Section 4.4 or 4.5 to compute the MMS value of every agent approximately up to a factor $(1-\epsilon/2)$. If $\overline{\mathsf{MMS}}_i$ is the value returned by the algorithm for agent i, we know $\overline{\mathsf{MMS}}_i \geq \min\{(1-\epsilon/2)\mathsf{MMS}_i, (1/(1-\epsilon/2))\mathsf{MMS}_i\}$. The algorithm then tries to find an $(\alpha-\epsilon/2)^+$-$\overline{\mathsf{MMS}}_i$ allocation, and fails only when an α-MMS allocation does not exist.

High-level Approach. At a high level, the algorithm to find an $(\alpha-\epsilon/2)^+$-$\overline{\mathsf{MMS}}_i$ allocation is as follows. We will classify all items as BIG, based on if they are highly valued by any agent relative to her MMS value, or SMALL otherwise. Although the MMS values of agents can be arbitrarily small, we show that the number of BIG items is a function of n, hence constant from condition 1. Therefore, we can efficiently enumerate all partitions of the BIG items.

For each partition, we allocate the SMALL items by solving an LP and rounding its solution.

The LP ensures a fractional solution where every agent gets at least an α-$\overline{\mathsf{MMS}}$ valued bundle. Next, through a careful rounding, we show that if there is an α-MMS allocation where the BIG items are allocated according to the current partition, then the allocation of all items obtained after rounding the LP solution is $(\alpha - \epsilon/2)^+$-$\overline{\mathsf{MMS}_i}$. Among all the fractional α-MMS allocations found by combining some BIG item partition with the allocation of SMALL as per the LP solution, we find the one, say $\mathcal{A} = [A_1, \cdots, A_n]$, with the highest value for the sum of valuations of all the agents, i.e., $\sum_i v_i(A_i)$. That is, we find a fractional allocation,

$$\mathcal{A} \in \underset{B \in \Pi_n(\mathsf{BIG})}{\operatorname{argmax}} \max_{A_i \supseteq B_i, A \text{ is } \alpha\text{-MMS}} \sum_{i \in \mathcal{N}} v_i(A_i).$$

Finally, we show that the rounded solution, call it $\mathcal{A}^r$, is γ-PO, by showing that for an allocation to γ-Pareto dominate $\mathcal{A}^r$, it must be an α-MMS allocation and have higher welfare than $\mathcal{A}$. This proof is quite involved and uses several new ideas, including the way we round the LP solution, to show Pareto optimality of the integral allocation (Recall from [75] that testing PO is coNP-hard, and market equilibrium is the only non-trivial technique in all known literature [allocations with highest sum of valuations are trivially PO] to certify that an allocation is PO). The following bound on the MMS_i values will be useful in the analysis, and follows from Lemma 4.1.2 and $|v_i(\mathcal{M})| = n,\ \forall i \in \mathcal{N}$.

Lemma 4.2.1. *For each agent $i \in \mathcal{N}$, $\mathsf{MMS}_i \leq 1$ if $v(\mathcal{M}) \geq 0$, otherwise $\mathsf{MMS}_i \leq -1$.*

In the remaining section, we will discuss the details and formalize these ideas. For brevity, at times we will refer to $\overline{\mathsf{MMS}_i}$ as $\tilde{\mu}_i$ and to MMS_i as μ_i.

4.2.1 BIG and SMALL Items

Next we classify items into sets BIG and SMALL, and show bounds on the size of the BIG items set.

Definition 4.2.1 (Big and Small items)**.** *The sets of all* BIG *goods* (BIG_i^+) *and* BIG *chores* (BIG_i^-) *of agent i are defined as,*

$$\mathsf{BIG}_i^+ := \{j \in \mathcal{M}^+ \mid (\tilde{\mu}_i \geq 0 \text{ and } v_{ij} > \epsilon\tilde{\mu}_i/(2n)) \text{ or } (\tilde{\mu}_i < 0 \text{ and } v_{ij} > \epsilon/(2n))\}, \text{ and}$$
$$\mathsf{BIG}_i^- := \{j \in \mathcal{M}^- \mid -v_{ij} > \epsilon/(2n)\}.$$

The union of all sets BIG_i^+ is called $\mathsf{BIG}^+ = \cup_i \mathsf{BIG}_i^+$, and of all BIG_i^- sets is called $\mathsf{BIG}^- = \cup_i \mathsf{BIG}_i^-$. Finally, the set of all BIG *items is called* $\mathsf{BIG} := \mathsf{BIG}^+ \cup \mathsf{BIG}^-$.
Any item that is not in BIG *is called a* SMALL *item. We define* SMALL *goods and chores for agent i as $\mathsf{SMALL}_i^+ = \mathcal{M}_i^+ \backslash \mathsf{BIG}_i^+$, and $\mathsf{SMALL}_i^- = \mathcal{M}_i^- \backslash \mathsf{BIG}_i^-$. Similarly, the sets of* SMALL

goods, SMALL *chores, and* SMALL *items are respectively* $\mathsf{SMALL}^+ = \mathcal{M}^+ \backslash \mathsf{BIG}^+$, $\mathsf{SMALL}^- = \mathcal{M}^- \backslash \mathsf{BIG}^-$, *and* $\mathsf{SMALL} = \mathsf{SMALL}^+ \cup \mathsf{SMALL}^-$.

In the remaining section, we will show the size of BIG is constant. For this, we make two useful observations, then show the bound on BIG.

Claim 4.2.1. *For the approximate* MMS *values* $\tilde{\mu}_i$*, we have, if* $\mu_i > 0$*, then* $\tilde{\mu}_i \in [(1-\epsilon/2)\mu_i, \mu_i]$*, if* $\mu_i = 0$ *then* $\tilde{\mu}_i = 0$ *and if* $\mu_i < 0$*, then* $\tilde{\mu}_i \in [\mu_i/(1-\epsilon/2), \mu_i]$.

Proof. We know from condition 2 of the α-MMS + PO problem that $v_i(\mathcal{M}) \geq \tau \cdot \min\{v_i^+, v_i^-\}$. After scaling, we have $|v_i(\mathcal{M})| = n$. For agents where $v_i(\mathcal{M}) \geq 0$, $n \geq \tau \cdot v_i^- \Rightarrow v_i^- \leq n/\tau = O(n)$. Also, $n = v_i^+ - v_i^- \Rightarrow v_i^+ \leq n(1+1/\tau) = O(n)$. Analogously we prove the claim when $v_i(\mathcal{M}) < 0$. QED.

Next, recall the definitions of v_i^+ and v_i^- from equation (4.2). The next claim follows from condition 2 of the problem.

Claim 4.2.2. *For all agents* i*,* $v_i^+ \leq O(n), v_i^- \leq O(n)$.

Next lemma shows a bound on $|\mathsf{BIG}|$. For this we show the bound of $O(n^2/\epsilon)$ on $|\mathsf{BIG}_i^+|$ and $|\mathsf{BIG}_i^-|$ for each agent i. Note that if $\tilde{\mu}_i$ is big enough then it is easy to prove that $|\mathsf{BIG}_i^+|$ is a constant. The difficulty is when $\tilde{\mu}_i$ is arbitrarily small, in which case $|\mathsf{BIG}_i^+|$ can potentially be large – a tricker case. The bound on $|\mathsf{BIG}_i^-|$ follows from the definition of BIG_i^- together with Claim 4.2.2.

Lemma 4.2.2. *The number of big items, i.e.,* $|\mathsf{BIG}| \leq O(n^3/\epsilon)$.

Proof. Since $\mathsf{BIG} = \cup_{i\in\mathcal{N}} \mathsf{BIG}_i^+ \cup_{i\in\mathcal{N}} \mathsf{BIG}_i^-$, to prove the lemma it suffices to show that the number of BIG goods and chores of every agent $i \in \mathcal{N}$ is $|\mathsf{BIG}_i^+|, |\mathsf{BIG}_i^-| \leq O(n^2/\epsilon)$. Fix an agent $i \in \mathcal{N}$. First we will show bound on $|\mathsf{BIG}_i^+|$.
Case 1: $\tilde{\mu}_i \geq 0$.

$$\text{If } \tilde{\mu}_i \geq 1/3, |\mathsf{BIG}_i^+| \leq |\{j \in \mathcal{M} : v_{ij} > \epsilon/(6n)\}| \leq \frac{v_i^+}{\epsilon/(6n)} \leq \frac{6n}{\epsilon} O(n) = O(n^2/\epsilon) \tag{4.3}$$

The last inequality follows by Claim 4.2.2. Otherwise, if $\tilde{\mu}_i < 1/3$, then $\mu_i \leq \tilde{\mu}_i/(1-\epsilon/2) < 1/(3-3\epsilon/2) < 2/3$. Divide BIG_i^+ into two sets as follows.

$$\mathsf{BIG}_i^+ = \{j : v_{ij} > \epsilon/(6n)\} \cup \{j : \epsilon\tilde{\mu}_i/(2n) < v_{ij} \leq \epsilon/(6n)\}. \tag{4.4}$$

Let us call the first set in Equation (4.4) LARGE and the second set MEDIUM. Similarly as for the case of $\tilde{\mu}_i \geq 1/3$, we can prove the size of LARGE is at most $O(n^2/\epsilon)$. We now prove that

the number of items in MEDIUM is at most $\frac{2n(n-1)}{(1-\epsilon/2)\epsilon} + (n-2)$. We show that if this is not true then there is a partition of all the items where all parts have value strictly more than μ_i for agent i which is a contradicts that μ_i being her MMS value. The partition is as follows. Add all items except the goods from MEDIUM to the first bundle. If i's value for this bundle is more than 1, divide MEDIUM to make $n-1$ bundles with at least $2n/((1-\epsilon/2)\epsilon)+1$ items in each. Then all the remaining bundles have value at least,

$$\left(\frac{2n}{(1-\epsilon/2)\epsilon}+1\right)\left(\frac{\epsilon}{2n}\right)\tilde{\mu}_i > \frac{\tilde{\mu}_i}{(1-\epsilon/2)} \geq \mu_i.$$

If the value of the first bundle is less than 1 for i, then we add enough goods from MEDIUM to each bundle one by one (first bundles and all remaining empty bundles) so that their value is at least $2/3 > \mu_i$. Since every item in MEDIUM has value at most $\epsilon/(6n)$, the value of the each bundle is less than $2/3$ before adding the last item and less than $2/3 + \epsilon/(6n) < 1$ later. As each bundle's value is at most 1 and $v(\mathcal{M}) = n$, there are enough items to make $(n-1)$ bundles, each of value at least $2/3$ which is greater than μ_i. This is a contradiction to definition of MMS_i.

Therefore, $|\mathsf{MEDIUM}| \leq \frac{2n(n-1)}{(1-\epsilon/2)\epsilon} + (n-2) = O(n^2/\epsilon)$. Hence $|\mathsf{BIG}_i^+| = |\mathsf{LARGE}| + |\mathsf{MEDIUM}| = O(n^2/\epsilon)$, for any i with $\tilde{\mu}_i \geq 0$.

Case 2: $\tilde{\mu}_i < 0$. Then by the definition of a BIG good for this case, $|\mathsf{BIG}_i^+| \leq v_i^+/(\epsilon/(2n)) = (2n/\epsilon)O(n) = O(n^2/\epsilon)$.

Next we show the bound on $|\mathsf{BIG}_i^-|$. By definition of a BIG chore, $|\mathsf{BIG}_i^-| \leq 2n \cdot v_i^-/\epsilon = O(n^2/\epsilon)$, as from Claim 4.2.2 we have $v_i^- \leq O(n)$. QED.

4.2.2 LP for Allocating SMALL Items, and Rounding

Given a partition $B^\pi = (B_1, \ldots, B_n)$ of BIG items, next we write an LP to find a *fractional* allocation of SMALL items such that together with B^π this allocation gives at least α-$\overline{\mathsf{MMS}}$ value to every agent. If there exists an α-MMS allocation where the BIG items are allocated as per B^π then we show that the LP has to be feasible.

For every agent i, denote by c_i the value from SMALL that i needs for her bundle's value to be

at least $\alpha \cdot \tilde{\mu}_i$ if $\tilde{\mu}_i \geq 0$ or $(1/\alpha) \cdot \tilde{\mu}_i$ otherwise, i.e., $c_i = \min\{(1/\alpha)\tilde{\mu}_i, \alpha\tilde{\mu}_i\} - v_i(B_i)$.

$$\max \quad \sum_{i \in \mathcal{N}} \left(\sum_{j \in \mathsf{SMALL}_i^+} v_{ij} x_{ij} - \sum_{j \in \mathsf{SMALL}_i^-} |v_{ij}| x_{ij} \right) \tag{4.5}$$

$$\text{s.t.} \quad \sum_{j \in \mathsf{SMALL}_i^+} v_{ij} x_{ij} - \sum_{j \in \mathsf{SMALL}_i^-} |v_{ij}| x_{ij} \geq c_i, \qquad \forall i \in \mathcal{N} \tag{4.6}$$

$$\sum_{i \in \mathcal{N}} x_{ij} \leq 1, \qquad \forall j \in \mathsf{SMALL}^+ \tag{4.7}$$

$$\sum_{i \in \mathcal{N}} x_{ij} \geq 1, \qquad \forall j \in \mathsf{SMALL}^- \tag{4.8}$$

$$x_{ij} \geq 0, \qquad \forall i \in \mathcal{N}, j \in \mathcal{M}. \tag{4.9}$$

We now prove two properties (Lemmas 4.2.3 and 4.2.4) that will help in obtaining an integral $(\alpha - \epsilon/2)$-$\overline{\mathsf{MMS}}$ allocation of items from a fractional α-$\overline{\mathsf{MMS}}$ allocation. Let us assume the LP has a solution, say $x = [x_{ij}]_{i \in \mathcal{N}, j \in \mathsf{SMALL}}$. We define a bipartite graph, called the *allocation graph*, corresponding to x as follows. There is a vertex corresponding to each agent in $\mathcal{N}$ in one part of vertices, and to each item in SMALL in the other part, and for all $i \in \mathcal{N}$ and $j \in \mathsf{SMALL}$, edge (i, j) exists if $x_{ij} > 0$. We show the following property of the allocation graph.

Lemma 4.2.3. *The allocation graph of* any *LP solution x can be made acyclic in such a way that in the allocation corresponding to the new graph, say $x' = [x'_{ij}]_{i \in \mathcal{N}, j \in \mathsf{SMALL}}$, every agent receives a bundle of the same or better value as in x.*

To prove the lemma, we show re-allocations can be done along any cycle in a certain way without any agent losing any value that eliminates at least one edge. For every cycle, we define a particular scaled valuation function, and define weights for the edges to reflect the values to agents from the adjacent items. Then we add and subtract weights in a certain way along the cycle, taking into consideration if the adjacent item is a good or a chore, so that the allocation corresponding to the new weights, or equivalently (scaled) values to agents, does not contain this cycle.

Proof. First we show how to eliminate one cycle, say C, in the allocation graph of x. That is, we define a new allocation x', that removes one cycle without reducing the value of any agent. Let there be k agents and k items in C, with the edges as,

$$a^1 - o^1 - a^2 - o^2 - \cdots - o^{i-1} - a^i - o^i - a^{(i+1)} \cdots - o^{k-1} - a^k - o^k - a^1.$$

Each agent a^i is partially assigned items o^i and o^{i-1} (by setting $0 \equiv k$) and each item o^i is partially assigned to agents a^i and a^{i+1} (by setting $k + 1 \equiv 1$). Without loss of generality, we may assume that items in C are solely considered as either a good or a chore by both agents sharing them,

otherwise, we can break the cycle by allocating the share of the other agent for this item to the one who considers it as a good. We call an item a good if both the agents sharing it consider it so, else a chore.

First we argue the case when there is at least one good. Without loss of generality we assume o^k is a good. Let $X^C = [x_{11}, x_{21}, x_{22}, \ldots, x_{k(k-1)}, x_{kk}, x_{1k}]$ be the allocation vector of cycle C. Also let $\tilde{V}^C = [\tilde{v}_{11}, \tilde{v}_{21}, \tilde{v}_{22}, \ldots, \tilde{v}_{k(k-1)}, \tilde{v}_{kk}, \tilde{v}_{1k}]$ be the vector representing the *scaled* values of the agents for the items assigned to them in C, defined as,

$$\tilde{v}_{ij} = \begin{cases} v_{ij} & \text{if } i = 1 \\ v_{ij}\left(\frac{v_{(i-1)(i-1)}}{v_{i(i-1)}}\right) & \text{otherwise.} \end{cases}$$

In $\tilde{V}^C$ the valuations of the agents are scaled in a way so that agents sharing an item in C have the same value for that item (except for item k). Without loss of generality we assume $\tilde{v}_{1k} \leq \tilde{v}_{kk}$. Let $U^C = [u_{11}, u_{21}, u_{22}, \ldots, u_{k(k-1)}, u_{kk}, u_{1k}]$ be the utility vector of C where $u_{ij} = \tilde{v}_{ij}x_{ij}$. Let δ be the minimum of smallest positive $u_{ij}, i \neq j$ (even indexes of U^C) and smallest $|u_{ii}|, u_{ii} < 0$ (odd indexes of U^C). Define

$$u'_{ij} := \begin{cases} u_{ij} - \delta & \text{if } i \neq j \\ u_{ij} + \delta & \text{otherwise.} \end{cases}$$

Then the desired x' is defined as,

$$x'_{ij} := \begin{cases} u'_{ij}/\tilde{v}_{ij} & \text{if } i, j \in C \\ x_{ij} & \text{otherwise.} \end{cases}$$

By choice of δ, at least one x'_{ij} with $x_{ij} > 0$ will be 0 and no new edge is added to the allocation graph so the cycle C is removed. We need to show that the new x'_{ij}'s present a feasible allocation. By choice of δ, we can see that $u'_{ij} \geq 0$ when j is a good for agents sharing it in C, $u'_{ij} \leq 0$ otherwise. For all agents a^i in C, $u_{i(i-1)} + u_{ii} = u'_{i(i-1)} + u'_{ii}$ (by setting $1 - 1 = k$ for agent a^1) so each agent will get the same utility before removing the cycle. Also, we have $u_{ii} + u_{i(i+1)} = u'_{ii} + u'_{i(i+1)}$ and since for all items $O^i, i \in [k-1]$, $\tilde{v}_{ii} = \tilde{v}_{i(i+1)}$ we have $x_{ii} + x_{i(i+1)} = x'_{ii} + x'_{i(i+1)}$. For item k we have,

$$\begin{aligned} u'_{kk} = u_{kk} + \delta \implies x'_{kk} &= x_{kk} + \frac{\delta}{\tilde{v}_{kk}} \\ u'_{1k} = u_{1k} - \delta \implies x'_{1k} &= x_{1k} - \frac{\delta}{\tilde{v}_{1k}} \\ \implies x'_{kk} + x'_{1k} &\leq x_{kk} + x_{1k}\ . \end{aligned}$$

The last inequality hold because $\tilde{v}_{1k} \leq \tilde{v}_{kk}$. Therefore, all agents receive the same utility in the new allocation. But there may be an extra amount of good k available; we assign it to the agent who has the highest share of good k.

If all items in C are chores, we define $\tilde{V}$ and U similarly as for the previous case. Without loss of generality, we assume $\tilde{v}_{1k} \leq \tilde{v}_{kk}$ and we choose δ to be the smallest $|u_{ii}|, u_{ii} < 0$ (odd indexes of U^C). With the same analysis we get $u_{ii} + u_{i(i+1)} = u'_{ii} + u'_{i(i+1)}$ for all agents i, $x_{ii} + x_{i(i+1)} = x'_{ii} + x'_{i(i+1)}$ for items $i \neq k$ and $x'_{kk} + x'_{1k} \geq x_{kk} + x_{1k}$. Therefore, agents get the same utility with an extra amount of chore k assigned to some agent. We improve the utility of the agent who gets this share of chore k by reducing her share from chore k by making, $\sum_{i \in \mathcal{N}} x_{ik} = 1$.

We repeat this process for every cycle, removing at least one edge with every removal. Hence, in polynomial time, we get an acyclic allocation graph. QED.

The next lemma follows since an undirected, acyclic graph forms a tree.

Lemma 4.2.4. *The number of shared items in any acyclic allocation graph is at most* $n - 1$.

Proof. Suppose there are k shared goods. Consider the subgraph of the allocation graph with the $n + k$ nodes corresponding to all the buyers and only the shared goods. As this graph is acyclic, there are at most $n + k - 1$ edges. Further, each item is shared, meaning there are at least two edges incident to each node representing a good. Thus, there are at least $2k$ edges. The inequality $n+k-1 \geq 2k$ is satisfied only when $k \leq n-1$, hence there are at most $n-1$ shared goods. QED.

Next we describe the notion of envy graph from [106], a directed graph corresponding to any allocation, that will be used to round the LP solutions.

Envy Graph and Cycle Elimination. Given a set of agents $\mathcal{N}$ and an *integral* allocation $\mathcal{A}$ of a set of items among them, each node in the graph corresponds to an agent in $\mathcal{N}$. There is a directed edge $(i \to k)$ corresponding to agents i and k if agent i values agent k's allocation more than her own. It is shown in [106] that the allocation can be modified so that its corresponding envy graph is acyclic, and no agent's valuation decreases. This is done by giving each agent in a cycle the bundle of her successor. The graph is updated and the process repeated until all cycles are eliminated. This process can be done efficiently, as shown in [106].

Claim 4.2.3. *In an allocation of* $\mathcal{M}$ *among* n *agents, every sink agent* i *corresponding to an acyclic envy graph has value at least* 1 *for her own bundle if* $v_i(\mathcal{M}) > 0$, *and at least* -1 *otherwise.*

Rounding the LP. Using Lemmas 4.2.3 and 4.2.4, we first modify the allocation graph of the LP solution so that it is a forest graph with at most $n-1$ shared items. Let S be the set of all the shared items, $S^{-\epsilon}$ the set of all the shared chores whose absolute value is more than $\epsilon|\tilde{\mu}_i|/(2n)$ for at least one agent, that is, $S^{-\epsilon} := \{j \in S \mid \exists i \in \mathcal{N}, |v_{ij}| > \epsilon|\tilde{\mu}_i|/(2n)\}$, and $S^{+} := S \setminus S^{-\epsilon}$. Allocate each

item j in S^+ to any agent i in $\text{argmax}_i v_{ij}$. Then consider the envy graph corresponding to this allocation of $\mathcal{M}\backslash S^{-\epsilon}$, and modify the allocation by eliminating all the cycles in the envy graph. Allocate all the items in $S^{-\epsilon}$ to a sink agent in the acyclic envy graph, and denoted it as i^t.

The following claim will be useful in proving the final allocation of the algorithm is γ-PO.

Claim 4.2.4. *If $S^{-\epsilon} \neq \emptyset$ then there exists an $i \in \mathcal{N}$ such that $v_i(\mathcal{M}) > 0$.*

Proof. Every agent with $v(\mathcal{M}) < 0$ has $\mu \leq -1$, from Lemma 4.2.1. The value of any chore in SMALL for any such agent is at most $\epsilon/2n \leq \epsilon|\mu|/2n \leq \epsilon|\tilde{\mu}|/2n$, as from Claim 4.2.1, $|\tilde{\mu}| \geq |\mu|$. Hence, if $S^{-\epsilon} \neq \emptyset$, then the agent who values any item in $S^{-\epsilon}$ more than $\epsilon\tilde{\mu}/(2n)$ has $v(\mathcal{M}) > 0$. QED.

Finally, we show the maximum loss in value of each agent in the rounding process, which will be used to ensure that the algorithm returns an $(\alpha - \epsilon)^+$-MMS allocation.

Lemma 4.2.5. *In the rounding process, i^t loses at most $\epsilon/2$ value and every other agent i loses at most $\epsilon|\tilde{\mu}_i|/2$ value.*

Proof. Every agent except i^t, in the worst case, loses all her shared goods and gains all her shared chores in S^+, and has no shared chores in $S^{-\epsilon}$, as she only gains from the rounding of items in $S^{-\epsilon}$. Her maximum loss from the items in S^+ is at most $(n-1)\cdot\epsilon\tilde{\mu}/(2n) \leq \epsilon\tilde{\mu}/2$, as $|S^+| \leq |S| \leq n-1$, from Lemma 4.2.4. For agent i^t, her loss from S in the worst case is at most $(n-1)\cdot\epsilon/(2n) \leq \epsilon$, as each item in S has absolute value at most $\epsilon/(2n)$ for her. QED.

4.2.3 PTAS for α-MMS+PO

Algorithm 4 combines the ideas in the previous sections and finds an $(\alpha - \epsilon)^+$-MMS allocation if an α-MMS allocation exists, else returns an empty allocation to indicate that no α-MMS allocation exists. The algorithm works as follows. First, it finds the approximate MMS values of all agents using the algorithms from Section 4.4 and Electronic Companion Section 4.5, and classifies all items as BIG or SMALL. Then among all allocations of BIG items where the corresponding LP has a solution, if any, it finds the combined allocation of BIG and SMALL, called $\mathcal{A}$, with the highest social welfare where SMALL may be fractionally allocated.

From constraint 4.6 of the LP, $\mathcal{A}$ is α-MMS, and as shown in Lemma 4.2.6, its rounded allocation, denoted as $\mathcal{A}^r$, is $(\alpha - \epsilon)^+$-MMS. To ensure $\mathcal{A}^r$ is also γ-PO, for technical reasons we require the sum of absolute values of all agents to be at least α whenever there is at least one agent with $v_i(\mathcal{M}) > 0$. If $\mathcal{A}^r$ does not satisfy this condition, Algorithm 4 ensures a stronger guarantee, namely at least one agent values her own bundle at least 1, by modifying $\mathcal{A}^r$ as follows. Let $\mathcal{N}^+$ be the set of agents with $v_i(\mathcal{M}) > 0$. Note that, for an $i \in \mathcal{N}^+$, $v_i(\mathcal{M}) = n$ but $v_i(A_i^r) < \alpha$, and hence there exists an agent $k \neq i$ such that i values k's bundle more than 1. We consider two

cases based on if k is in $\mathcal{N}^+$ or not. If $k \notin \mathcal{N}^+$, then k's MMS value is negative, thus even if we re-allocate her bundle to i and give k nothing (lines 18-19), her $(\alpha-\epsilon)^+$-MMS guarantee is maintained. If no such (i,k) pair is found, then we go to the other case, where k has to be given something. For this (lines 21-22), we construct a graph on $\mathcal{N}^+$ where there is an edge from i to k if $v_i(A_i^r) < \alpha$ and $v_i(A_k^r) \geq 1$. This graph has to have a cycle (See proof of Claim 4.2.6), and swapping bundles along the cycle gives value more than 1 to every agent along the cycle.

We will prove the correctness of the algorithm in the remaining section. In what follows, we denote by $\mathcal{A} = [\mathcal{A}_1 \cdots, \mathcal{A}_n]$ and $\mathcal{A}^r = [\mathcal{A}_1^r \cdots, \mathcal{A}_n^r]$ respectively the fractional allocation that is rounded after Line 12 and its rounded allocation. First we show that the algorithm returns an $(\alpha-\epsilon)^+$-MMS allocation if an α-MMS allocation exists.

Lemma 4.2.6. *If the LP has a solution for any partition of* BIG*, then $\mathcal{A}^r$ is an $(\alpha-\epsilon)^+$-MMS allocation.*

Proof. First, we argue for the allocation obtained after the rounding step on Line 14. Consider agent i^t. Since i^t corresponds to a sink node in the envy graph, from Claim 4.2.3 and Lemmas 4.2.5 and 4.2.1, her value for her bundle in $\mathcal{A}^r$ is at least $1-\epsilon/2 \geq 1-\epsilon \geq (\alpha-\epsilon)\mu_{i^t}$ if $\tilde{\mu}_{i^t} \geq 0$, and $-1-\epsilon/2 \geq -1-\epsilon \geq (1+\epsilon)\mu_{i^1} \geq \frac{1}{(\alpha-\epsilon)}\mu_{i^1}$ otherwise. Next, every agent i except i^t, according to constraint (4.6) of the LP, receives a bundle of value at least c_i from SMALL in the fractional allocation of SMALL corresponding to $\mathcal{A}$. Thus, for all $i \neq i^t$, their value for their bundle in $\mathcal{A}^r$ is at least $v_i(B_i) + c_i - n\epsilon \cdot |\tilde{\mu}_i|/(2n) \geq \min\{(1/\alpha)\tilde{\mu}_i, \alpha\tilde{\mu}_i\} - \epsilon \cdot |\tilde{\mu}_i|/2$, from Lemma 4.2.5 and by definition of c_i. Combined with Claim 4.2.1, when $\tilde{\mu}_i \geq 0$, this is at least $(\alpha-\epsilon/2)\tilde{\mu}_i \geq (\alpha-\epsilon/2)(1-\epsilon/2)\mu_i \geq (\alpha-\epsilon)\mu_i$. When $\tilde{\mu}_i < 0$, this value is at least $\frac{1}{\alpha}\tilde{\mu}_i + \epsilon\tilde{\mu}_i/2$. As $(\frac{1}{\alpha}+\epsilon/2) \leq \frac{1}{(\alpha-\epsilon/2)}$, and $\tilde{\mu}_i < 0$, along with Claim 4.2.1, $(\frac{1}{\alpha}+\epsilon/2)\tilde{\mu}_i \geq \frac{1}{(\alpha-\epsilon/2)(1-\epsilon/2)}\mu_i \geq \frac{1}{(\alpha-\epsilon)}\mu_i$.

Any further modifications of the allocation can occur when (a) there is a pair of agents i,k with $v_i(\mathcal{A}_k^r) > 1$ and $v_k(\mathcal{M}) < 0$, or when (b) there is a cycle of agents with value at most α for their own bundle and value at least 1 for the next. The only agents whose value decreases in these steps are those with $v_k(\mathcal{M}) < 0$, who after the swap receive no item. As $v_k(\mathcal{M}) < 0$, then $\mu_k < 0$, hence they receive at least $\mu_k \geq \frac{1}{(\alpha-\epsilon)}\mu_k$ valued bundle. As no other agents lose, they still retain an $(\alpha-\epsilon)^+$-MMS bundle. QED.

Note that, the steps after rounding maintains the MMS guarantee. Since by construction, the LP has to be feasible whenever BIG items are allocated as per an α-MMS allocation, we get as a corollary,

Corollary 4.2.1. *If an α-MMS allocation exists, Algorithm 4 returns an $(\alpha-\epsilon)^+$-MMS allocation.*

Proof. If an α-MMS allocation exists, then for the partition of BIG corresponding to this allocation, say $B^\pi = [B_1 \cdots, B_n]$, there is an integral allocation of SMALL where every agent i gets value

Algorithm 4: $(\alpha - \epsilon)^+$-MMS + γ-PO allocation of mixed items to non-identical agents

Input : Instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$, $\alpha \in (\epsilon, 1]$, $\gamma > 0$ and $\epsilon > 0$
Output: $(\alpha - \epsilon)^+$-MMS + γ-PO allocation or report α-MMS allocation does not exist

$\epsilon \leftarrow \min\{\epsilon, \frac{\gamma\alpha}{(1+\gamma)}\}$, $flag \leftarrow false$
For all $i \in \mathcal{N}$, $\mathcal{A}^r_i \leftarrow \emptyset$ `// initialize` $\mathcal{A}^r$ `as the empty allocation`
$\mathcal{A} \leftarrow$ lowest social welfare allocation, i.e., give every item j to agent i with smallest v_{ij}
For all i, $\overline{\mathsf{MMS}_i} \leftarrow (1 - \frac{\epsilon}{2})$-$\mathsf{MMS}_i$ value of agent i `// use Algorithm from Section 4.4`
Define BIG and SMALL according to Definition 4.2.1
for *all allocations* $B^\pi = [B_1, B_2 \ldots, B_n]$ *of* BIG **do**
    Solve the LP (Equations (4.5)-(4.9)) for allocating SMALL items
    if *LP has a solution* **then**
        $flag \leftarrow true$
        $A^\pi \leftarrow$ Allocation of SMALL in optimal LP combined with B^π
        $\mathcal{A} \leftarrow$ Allocation from $(\mathcal{A}, A^\pi)$ with higher welfare, i.e., $\sum_i v_i(\mathcal{A}_i)$
if $flag = true$ **then**
    Make allocation graph of $\mathcal{A}$ acyclic using Lemma 4.2.3
    Round off $\mathcal{A}$ and obtain $\mathcal{A}^r$ by applying the rounding method from Section 4.2.2
    if $\sum_i |v_i(\mathcal{A}^r_i)| < \alpha$, *and* $\exists i : v_i(\mathcal{M}) > 0$ `// technical steps to get` γ–PO **then**
        $\mathcal{N}^+ \leftarrow$ set of agents i with $v_i(\mathcal{M}) > 0$
        if $\exists i \in \mathcal{N}^+,\ k \notin \mathcal{N}^+ :\ v_i(\mathcal{A}^r_k) \geq 1$ **then**
            modify $\mathcal{A}^r$ by giving $\mathcal{A}^r_k$ to i, and giving k nothing `// note: k has negative MMS`
        else
            Construct the following directed graph $G(V, E)$. $V = \mathcal{N}^+$, directed edge $(i \to k) \in E$ if $v_i(\mathcal{A}^r_i) < \alpha$ and $v_i(\mathcal{A}^r_k) \geq 1$ `// G has a cycle as for all` $i \in V :\ v_i(\mathcal{A}^r_i) < \alpha \leq 1$
            Swap bundles along any 1 cycle in G by giving every agent her successor's bundle
return $\mathcal{A}^r$

$\alpha \cdot \tilde{\mu}_i - v_i(B_i) \geq c_i$ from SMALL. Thus, the LP will have a (fractional) solution. From Lemma 4.2.6, the resulting allocation obtained by rounding the LP solution is $(\alpha - \epsilon)^+$-MMS. QED.

Finally, we show the approximate Pareto optimality of the $(\alpha - \epsilon)^+$-MMS allocation returned. This is the more involved part. For this, we use the notion of *social welfare* of any allocation, defined as the sum of values of all agents. Formally, for an allocation $A = [A_1 \cdots, A_n]$ among n agents, define its social welfare as $w(A) = \sum_i v_i(A_i)$.

Lemma 4.2.7. *If an α-MMS allocation exists, then $\mathcal{A}$ has the highest welfare among all the α-MMS allocations of $\mathcal{M}$ among $\mathcal{N}$ obtained by allowing* SMALL *to be fractionally allotted.*

Proof. Let $\mathcal{S}^B \subseteq \Pi_n(\text{BIG})$ be the set of all partitions of BIG corresponding to which there is a fractional α-MMS allocation, or in other words, for which the LP has a solution. For every partition B^π in $\mathcal{S}^B$, the objective function of the LP ensures that the allocation of SMALL returned by the algorithm has the highest social welfare among all allocations that satisfy the LP constraints. Hence, among all α-MMS allocations where the partition of BIG is B^π, the allocation returned, say A^π, has the highest social welfare. Formally, let $\mathcal{S}^{A,B^\pi}$ be the set of all α-MMS allocations corresponding to the partition B^π. Then $A^\pi \in \text{argmax}_{A \in \mathcal{S}^{A,B^\pi}} \sum_i w(A)$.

Let the set of allocations A^π, one corresponding to each partition $B^\pi \in \mathcal{S}^B$, be $\mathcal{S}^A$. From Line 11 of the algorithm, $\mathcal{A}$ has the highest social welfare among all. Formally, $\mathcal{A}$ is in $\text{argmax}_{A \in \mathcal{S}^A}\{w(A)\}$. Combining with the above characterization of the allocations in $\mathcal{S}^A$, we have,

$$\mathcal{A} \in \max_{B^\pi \in \mathcal{S}^B} \operatorname*{argmax}_{A \in \mathcal{S}^{A,B^\pi}} \{w(A)\},$$

thus proving the lemma. QED.

Next, we prove two key properties (Lemmas 4.2.8 and 4.2.9) of any *integral* allocation that γ-Pareto dominates $\mathcal{A}^r$. Suppose $\mathcal{A}^*$ is such an allocation.

Lemma 4.2.8. *$\mathcal{A}^*$ is an integral α-MMS allocation.*

Proof. We will use the following relation between ϵ, α and γ. We have,

$$\epsilon \leq \frac{\gamma\alpha}{(1+\alpha)} \Rightarrow \epsilon \leq \gamma(\alpha - \epsilon) \Rightarrow \gamma \geq \frac{\epsilon}{(\alpha - \epsilon)} \Rightarrow (1+\gamma) \geq \frac{\alpha}{(\alpha - \epsilon)}. \tag{4.10}$$

We know that $\mathcal{A}^r$ is an $(\alpha - \epsilon)^+$-MMS allocation. Hence, agents i with $\mu_i \geq 0$ get a bundle of value at least $(\alpha - \epsilon)\mu_i \geq 0$ in $\mathcal{A}^r$, hence get a bundle of value at least $(1+\gamma)(\alpha - \epsilon)\mu_i$ in $\mathcal{A}^*$. From equation (4.10), this is at least $\alpha\mu_i$. Next, consider agents with $\mu_i < 0$. If they receive a bundle of positive value in $\mathcal{A}^r$, then they also receive a positive valued, hence a bundle of value more than $\mu_i \geq \frac{1}{\alpha}\mu_i$, in $\mathcal{A}^*$. And if they get a negative valued bundle of value at least $\frac{1}{(\alpha-\epsilon)}\mu_i$ in $\mathcal{A}^r$, then they

get a bundle of value at least $\frac{1}{(1+\gamma)(\alpha-\epsilon)}\mu_i$, which from equation (4.10) and the fact that $\mu_i < 0$, is at least $\frac{1}{\alpha}\mu_i$. Hence, $\mathcal{A}^*$ is an integral α-MMS allocation. QED.

The next property of $\mathcal{A}^*$ is that it will have higher social welfare than (fractional allocation) $\mathcal{A}$. To prove this, we first prove two technical claims.

Claim 4.2.5. *$w(\mathcal{A}^r) \geq w(\mathcal{A}) - \epsilon$ if $S^{-\epsilon} \neq \emptyset$, else $w(\mathcal{A}^r) \geq w(\mathcal{A})$.*

Proof. We consider each step in Algorithm 4 that changes the allocation from $\mathcal{A}$ to $\mathcal{A}^r$, and see how it changes the social welfare. The first step is making the allocation graph of $\mathcal{A}$ acyclic. Here every agent's value, hence the social welfare also remains the same. The next step is the rounding process. Here, first the items in S^+ are allotted to the agents with the highest value for them, hence the sum of values of the items from S^+ in $\mathcal{A}^r$ is at least as much as that in $\mathcal{A}$. As no other item's allocation changes, the social welfare from them remains the same. Hence this part only improves the social welfare. Then the envy graph cycle elimination only improves the value of every agent, hence does not reduce the social welfare. At this point, $w(\mathcal{A}^r) \geq w(\mathcal{A})$. If $S^{-\epsilon} = \emptyset$, the rounding process ends, hence this inequality holds, otherwise from Lemma 4.2.5, allocating the items from $S^{-\epsilon}$ reduces the sink agent's value, hence the social welfare, by at most ϵ, giving $w(\mathcal{A}^r) \geq w(\mathcal{A}) - \epsilon$. Now we show the next part of the algorithm does not reduce the social welfare of $\mathcal{A}^r$, hence these relations remain true, thus proving the claim.

First consider the if statement on Line 18. For agents i, k when k's bundle is given to i, only the allocation of items in $\mathcal{A}^r_k$ changes and the allocation of $\mathcal{M} \backslash \mathcal{A}^r_k$ remains the same. Now $v_k(\mathcal{A}^r_k) < \alpha$ and $v_i(\mathcal{A}^r_k) > 1$, otherwise this step would not be executed. Hence the social welfare changes by $v_i(\mathcal{A}^r_k) - v_k(\mathcal{A}^r_k) > 1 - \alpha \geq 0$. Finally, suppose some bundles are swapped along some cycle by executing Line 22. Every agent had value at most α for their bundle, and received a bundle of value at least 1. Thus, the value of every agent in the cycle changes by at least $1 - \alpha > 0$, and every other agent's bundle, hence its value, remains the same. Thus, the social welfare does not decrease in this step as well. QED.

Claim 4.2.6. *If there is an agent i with $v_i(\mathcal{M}) > 0$, then there exists some agent i' with $v_{i'}(\mathcal{A}^r_{i'}) \geq \alpha$.*

Proof. If the claim is true for the allocation obtained after rounding $\mathcal{A}$, then this is the allocation returned, hence we are done. Otherwise, the if condition of Line 15 is executed. If the condition of Line 15 is true, then as the allocation before this line was $(\alpha - \epsilon)^+$-MMS, $v_i(\mathcal{A}^r_i) \geq 0$, and after obtaining k's bundle, $v_i(\mathcal{A}^r_i \cup \mathcal{A}^r_k) \geq 0 + 1 = 1$.

Otherwise, for every agent i in $\mathcal{N}^+$ we have $v(\mathcal{M}) = n$, thus there is at least one agent $k \in \mathcal{N}^+$ such that $v_i(\mathcal{A}^r_k) \geq 1$. The graph $G(V, E)$ has an edge $i \to k$ in this case. As $\sum_i |v_i(\mathcal{A}^r_i)| < \alpha$, for every i we have $v_i(\mathcal{A}^r_i) < \alpha \leq 1$. Thus for every edge $i \neq k$. Hence, G has at least one cycle. After

swapping along a cycle, all agents along the cycle receive the bundle of their successor, hence have value at least 1 for their own bundle. Thus, the final allocation has some agent with value at least $1 \geq \alpha$ for her bundle. QED.

Lemma 4.2.9. *$\mathcal{A}^*$ has higher social welfare than $\mathcal{A}$.*

Proof. For cleaner exposition, we will denote $v_i(A_i)$ by $v_i(A)$ for any allocation A.

By definition of a γ-Pareto dominating allocation, the social welfare of allocation $\mathcal{A}^*$ is,

$$\begin{aligned} w(\mathcal{A}^*) &> (1+\gamma) \cdot \sum_{i:v_i(\mathcal{A}^r)\geq 0} v_i(\mathcal{A}^r) + \frac{1}{1+\gamma} \cdot \sum_{i:v_i(\mathcal{A}^r)<0} v_i(\mathcal{A}^r) \\ &= \sum_i v_i(\mathcal{A}^r) + \gamma \cdot \sum_{i:v_i(\mathcal{A}^r)\geq 0} v_i(\mathcal{A}^r) - \frac{\gamma}{(1+\gamma)} \cdot \sum_{i:v_i(\mathcal{A}^r)<0} v_i(\mathcal{A}^r) \\ &= w(\mathcal{A}^r) + \gamma \cdot \sum_{i:v_i(\mathcal{A}^r)\geq 0} v_i(\mathcal{A}^r) + \frac{\gamma}{(1+\gamma)} \cdot \sum_{i:v_i(\mathcal{A}^r)<0} |v_i(\mathcal{A}^r)|. \end{aligned} \tag{4.11}$$

Let $\mathbb{1}_{S^{-\epsilon}}$ be an indicator variable for whether $S^{-\epsilon}$ is non-empty. Substituting the relation between $w(\mathcal{A})$ and $w(\mathcal{A}^r)$ from Claim 4.2.5 in equation (4.11) we get, $w(\mathcal{A}^*) > w(\mathcal{A}) - \epsilon \cdot \mathbb{1}_{S^{-\epsilon}} + \gamma \cdot \sum_{i:v_i(\mathcal{A}^r)\geq 0} v_i(\mathcal{A}^r) + \frac{\gamma}{(1+\gamma)} \cdot \sum_{i:v_i(\mathcal{A}^r)<0} |v_i(\mathcal{A}^r)|$. Hence, to prove the lemma, it suffices to show,

$$\gamma \cdot \sum_{i:v_i(\mathcal{A}^r)\geq 0} v_i(\mathcal{A}^r) + \frac{\gamma}{(1+\gamma)} \cdot \sum_{i:v_i(\mathcal{A}^r)<0} |v_i(\mathcal{A}^r)| - \epsilon \cdot \mathbb{1}_{S^{-\epsilon}} \geq 0. \tag{4.12}$$

If $S^{-\epsilon} = \emptyset$, we are done, as all values on the left hand side in the above equation, say L, are non-negative. Hence, we now prove the equation when $S^{-\epsilon} \neq \emptyset$, that is, $\mathbb{1}_{S^{-\epsilon}} = 1$.

We know $\epsilon \leq \frac{\gamma\alpha}{(1+\gamma)}$, and as $\gamma \geq 0$, $\gamma \geq \frac{\gamma}{1+\gamma}$. Substituting these relations in L, we have,

$$L \geq \frac{\gamma}{1+\gamma} \cdot \sum_{i:v_i(\mathcal{A}^r)\geq 0} v_i(\mathcal{A}^r) + \frac{\gamma}{(1+\gamma)} \cdot \sum_{i:v_i(\mathcal{A}^r)<0} |v_i(\mathcal{A}^r)| - \frac{\gamma\alpha}{(1+\gamma)} = \frac{\gamma}{1+\gamma} \left(\sum_i |v_i(\mathcal{A}^r)| - \alpha \right).$$

But as $S^{-\epsilon} \neq \emptyset$, from Claim 4.2.4, there is some agent i with $v_i(\mathcal{M}) > 0$. Then from Claim 4.2.6, $v_{i'}(\mathcal{A}^r) \geq \alpha$ for at least one agent i'. Hence, $\frac{\gamma}{1+\gamma} \left(\sum_i |v_i(\mathcal{A}^r)| - \alpha \right) \geq 0$, thus proving equation (4.12), hence the lemma. QED.

Corollary 4.2.2. *If an α-MMS allocation exists, Algorithm 4 returns a γ-PO allocation.*

Proof. For contradiction, suppose $\mathcal{A}^r$ is not γ-PO. Then there is another allocation, say $\mathcal{A}^*$, that γ-Pareto dominates $\mathcal{A}^r$. From Lemmas 4.2.8 and 4.2.9, $\mathcal{A}^*$ is an integral α-MMS allocation with higher social welfare than $\mathcal{A}$. From Lemma 4.2.7, this is a contradiction. QED.

Using Corollaries 4.2.1 and 4.2.2, the next theorem obtains the main result.

Theorem 4.2.3. *Given an instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$ and constants $\alpha, \epsilon, \gamma > 0$, that is, an instance of the α-MMS + PO problem, Algorithm 4 returns an $(\alpha - \epsilon)^+$-MMS + γ-PO allocation if an α-MMS allocation exists, else reports it does not exist, in time $O(2^{O((n^3 \log n)/\min\{\epsilon^2, \gamma^2/2\})} m^3)$ where $n = |\mathcal{N}|$ is a constant. Thus, it is a PTAS.*

Proof. From Corollaries 4.2.1 and 4.2.2 the correctness of Algorithm 4 follows. Next we analyze the running time.

The time to compute the approximate MMS values is $O(n \cdot 2^{(n \log n)/\epsilon}(2^{1/\epsilon^2} n \log m + m))$, from the proofs of Theorems 4.4.1 and 4.5.1. Since $|\mathsf{BIG}| \leq O(n^3/\tau\epsilon)$ by Lemma 4.2.2, the number of iterations in the for loop enumerating all the allocations of the BIG items is $O(2^{O((n^3 \log n)/\epsilon)})$. Note that we re-define ϵ as $\min\{\epsilon, \frac{\alpha\gamma}{(1+\gamma)}\} \geq \min\{\epsilon, \frac{\gamma^2}{2}\} =: \zeta$, thus $|\mathsf{BIG}| \leq O(n^3/\zeta)$. Each iteration solves an LP of mn variables and $O(mn)$ constraints, hence takes time some polynomial function in (m, n) less than $O((mn)^3)$ [107]. Finding a cycle in the allocation graph requires time linear in the number of edges, at most $O(mn)$. Eliminating the cycle requires time $O(mn)$, and deletes at least one edge. Repeating the process until the graph is acyclic takes at most $O(mn)$ iterations, hence the making the allocation acyclic and rounding it steps take time at most $O(m^2n^2)$. Hence the total time for the algorithm in the worst case is,

$$O(n \cdot 2^{n \log n/\epsilon}(2^{1/\epsilon^2} n \log m + m)) + O(2^{O(n^3 \log n/\zeta)} m^3 n^3 + m^2 n^2) \leq O(2^{O((n^3 \log n)/\min\{\epsilon^2, \zeta\})} m^3),$$

$$= O(2^{O((n^3 \log n)/\min\{\epsilon^2, \gamma^2/2\})} m^3) = O(m^3),$$

as n, α, γ and ϵ are constant. QED.

4.2.4 PTAS for OPT-α-MMSPO

Our final goal is to solve OPT-α-MMS+PO problem, that is to find α-MMS+PO allocation for the highest possible α. In this section we design a PTAS for this problem: given $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$, and constants $\epsilon, \gamma > 0$, we design a polynomial-time algorithm to find $(\alpha - \epsilon)^+$-MMS + γ-PO allocation for the highest possible α. For this we will use the PTAS for α-MMS + PO problem described in the previous section.

Note that, for a given α, Algorithm 4 either returns $(\alpha - \epsilon)^+$-MMS+γ-PO allocation, or returns an *empty allocation*. And by Theorem 4.2.3, whenever it returns an empty allocation no α-MMS allocation exists. Using this, we run a simple binary search to find the highest value of $\alpha \in [\epsilon, 1]$ (up to a polynomial precision) for which Algorithm 4 returns a non-empty allocation. If empty allocation is returned for every α in our search, then we only need to ensure PO and therefore we return the social welfare maximizing allocation obtained by giving every item to the agent who values it the most.

We stop when the range of α values under consideration is $[c-\delta, c+\delta]$ for $\delta = \frac{1}{2^{poly(m)}}$ for some constant c, where $m = |\mathcal{M}|$. Clearly the number of iterations the binary search will take to get within such a range is at most $poly(m)$. Each iteration runs Algorithm 4 once, and hence finishes in $O(m^3)$ time (Theorem 4.2.3). Thus the overall running time of the algorithm is $poly(m)$, and the next theorem follows.

Theorem 4.2.4. *Given an instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$ and constants $\epsilon, \gamma > 0$, that is, an instance of the* OPT-α-MMS + PO *problem, there is a* PTAS *that runs for $(2^{O(1/\min\{\epsilon^2,\gamma^2\})}poly(m))$ time and returns an $(\alpha-\epsilon)^+$-MMS + γ-PO allocation such that for any $\alpha' > \alpha + \frac{1}{2^{m^c}}$, no α'-MMS allocation exists, where $c > 0$ is a constant.*

This completes the discussion of the OPT-α-MMS + PO problem. Next, we describe a PTAS for finding the MMS value of an agent for distributing a set of items $\mathcal{M}$ into n bundles according to a valuation function $v : \mathcal{M} \to \mathbb{R}$. We refer to this problem as the α-MMS problem with (n) identical agents. Using Lemma 4.1.1 we can find the sign of the MMS value. Hence, we describe two algorithms, one for each case when MMS ≥ 0 and otherwise. The following section discusses the algorithm for the former case.

4.3 NON-EXISTENCE OF α-MMS ALLOCATIONS

In this section, we show an instance for which there is no α-MMS allocation for any $\alpha > 0$. Our instance is a modification of the instance in [53] that shows that an MMS allocation in a goods only manna does not always exist. We take their exact instance, and add three chores to $\mathcal{M}$, each of absolute value equal to a small constant less than the agent's MMS values. For completeness, we discuss all details of the instance.

Let $\mathcal{N} = \{1,2,3\}$, $\mathcal{M}^+ = \{(j,k) : j \in [3], k \in [4]\}$, $\mathcal{M}^- = \{(1),(2),(3)\}$, and $\mathcal{M} = \mathcal{M}^+ \cup \mathcal{M}^-$ respectively be the set of agents, goods, chores, and all items. In order to define the valuations of the agents for each of these items, we first define matrices $O, E^{(1)}, E^{(2)}$, and $E^{(3)}$ as follows.

$$O = \begin{bmatrix} 17 & 25 & 12 & 1 \\ 2 & 22 & 3 & 28 \\ 11 & 0 & 21 & 23 \end{bmatrix}$$

$$E^{(1)} = \begin{bmatrix} 3 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad E^{(2)} = \begin{bmatrix} 3 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \quad E^{(3)} = \begin{bmatrix} 3 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

The valuation of each agent i for each good (j,k) is, $v_i(\{(j,k)\}) = 10^6 + 10^3 \cdot O_{jk} + E^{(i)}_{jk}$, and their value for each chore is -4054999.75.

From [53], every agent can divide all the goods in this instance into three bundles of value 4055000 each. Adding one chore to each of these makes every bundle's value 0.25. It can be verified that the average value of all items is 0.25 for every agent. As MMS cannot be higher than the average, the above allocation shows that every agent's MMS value is 0.25. [53] also show that there is no allocation of the goods were all agents get at least 4055000, and that the sum of any 3 goods is less than 4055000. As the values of goods are integers, every agent must get at least 4 goods for every chore in order to receive a positive valued bundle. If every agent is to get a positive valued bundle, the agent receiving less than 4055000 from the goods must not receive any chore, and must get at least one good. But then there are 3 chores and at most 11 goods remaining to be allotted. Hence, at least one agent will receive a negative valued bundle. Therefore, there is no allocation that can guarantee every agent a positive valued bundle, and the best α for which an α-MMS allocation exists is at most zero.

4.4 FINDING MMS VALUES OF AGENTS WHEN MMS ≥ 0

In this section we prove Theorem 4.2.2 for the case when MMS ≥ 0, i.e., given an instance $(n, \mathcal{M}, v)$, we describe an algorithm to find a $(1-\epsilon)$-MMS allocation for any constant $\epsilon > 0$. Using scale invariance (Lemma 4.1.3), here on we assume $v(\mathcal{M}) = n$ without loss of generality. Due to Lemma 4.1.2, this implies MMS $\leq v(\mathcal{M})/n = 1$.

The high level ideas used in the algorithm are as follows. First is a classification of all the items into two sets, Big and Small, based their value (Section 4.4.1). Using this we prove that |Big| is constant, which allows the enumeration of all allocations of Big, referred as *partitions* of Big to avoid confusion with allocations of $\mathcal{M}$. Next in Section 4.4.2 we explain a short procedure which allows us to characterize partitions of Big as *valid* or *invalid*. We show that there is at least one valid partition corresponding to which there is an MMS allocation, hence all invalid partitions can be discarded. Finally in Section 4.4.3, we describe a sub-routine called Bag-Fill, that greedily allocates or 'fills' the items from Small upon partitions or 'bags' of Big that satisfy certain constraints to obtain a $(1-\epsilon)$-MMS allocation. The main algorithm (Section 4.4.4) enumerates all partitions of Big, discards the invalid partitions, and applies Bag-Fill if its constraints are satisfied. If the constraints are not satisfied, we show that we can apply the PTAS for obtaining MMS allocations with identical agents for a goods manna, and obtain a $(1-\epsilon)$-MMS allocation.

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

4.4.1 Big and Small items

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

$$\mathsf{Big} = \{j \in \mathcal{M} : |v_j| \geq \tfrac{\epsilon}{2}\}.$$

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

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

Lemma 4.4.1. $|\mathsf{Big}| = O(n/\epsilon)$.

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

$$n \geq v(\mathcal{M}^+) - \frac{v(\mathcal{M}^+)}{1+\tau} \implies v(\mathcal{M}^+) \leq \frac{n(1+\tau)}{\tau}\,. \tag{4.13}$$

Finally, by the definition of Big^+ we have

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

Similarly, we have

$$n \geq (1+\tau)|v(\mathcal{M}^-)| - |v(\mathcal{M}^-)| \Rightarrow |v(\mathcal{M}^-)| \leq n/\tau. \tag{4.14}$$

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

As n and ϵ are constant, Lemma 4.4.1 implies that all partitions of Big can be enumerated in constant time.

4.4.2 Valid and Invalid partitions of Bigtems

Given an allocation $A^\pi = [A_1 \cdots, A_n]$ of $\mathcal{M}$, denote by $B^\pi = [B_1 \cdots, B_n]$ the allocation from $\mathcal{M} \setminus \mathsf{Small}^+$. Classify the bundles of A^π based on their value from B^π into sets $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$, together called the $\mathcal{B}$-sets, as follows.

$$\begin{aligned} \mathcal{B}_1 &:= \{A_k \in B^\pi : v(B_k) > 1\} & \mathcal{B}_2 &:= \{A_k \in B^\pi : 1 - \epsilon \leq v(B_k) \leq 1\} \\ \mathcal{B}_3 &:= \{A_k \in B^\pi : 0 \leq v(B_k) < 1 - \epsilon\} & \mathcal{B}_4 &:= \{A_k \in B^\pi : v(B_k) < 0\} \end{aligned} \tag{4.15}$$

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

Given a partition of Big B^π, we now explain a procedure using which we classify the partition as valid or invalid. First classify the bundles of B^π into four sets as per equation (4.15). Initially, all Small items are unallocated. Then while $\mathcal{B}_1 \neq \emptyset$ and $\mathsf{Small}^- \neq \emptyset$, assign any item from Small^- to any agent that has a bundle from $\mathcal{B}_1$. Re-classify the bundles using equation (4.15) and remove the assigned item from Small^- after every assignment. This procedure ends when either $\mathcal{B}_1$ or Small^- becomes empty, or both. If in the end $\mathcal{B}_1 \neq \emptyset$, and we also have (a) $\mathcal{B}_4 \neq \emptyset$ and (b) $v(\mathcal{B}_3 \cup \mathcal{B}_4 \cup \mathsf{Small}^+) < (1 - \frac{\epsilon}{2})(|\mathcal{B}_3| + |\mathcal{B}_4|)$ then we call B^π *invalid*. All partitions of Big that are not invalid are called valid.

Lemma 4.4.2. *There exists an* MMS *allocation where the* Big *items are allocated according to a valid partition.*

Proof. Let A^π be an MMS allocation with the lowest value of $|\mathcal{B}_1| + |\mathcal{B}_4|$, and let B^π be its corresponding partition of Big. Suppose B^π is invalid. Then the agents with bundles from $\mathcal{B}_3$ or $\mathcal{B}_4$ can only receive items from $\mathcal{B}_3 \cup \mathcal{B}_4 \cup \mathsf{Small}^+$ in A^π. From Lemma 4.1.2 and the definition of invalid partitions giving $v(\mathcal{B}_3 \cup \mathcal{B}_4 \cup \mathsf{Small}^+) < (1 - \frac{\epsilon}{2})(|B_3| + |B_4|)$, we have $\mu < (1 - \frac{\epsilon}{2})$.

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

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

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

4.4.3 Algorithm Bag-Fill

In this section we design the algorithm Bag-Fill (Algorithm 5) that generalizes algorithms in [49, 50, 51] to the mixed setting. Bag-Fill (Algorithm 5) takes as input an MMS instance $(n, \mathcal{M}, v)$, and a partition of the Big items of $\mathcal{M}$, denoted by $B^\pi = [B_1, B_2, \ldots, B_n]$ such that they satisfy one of the two condition sets (4.16) or (4.17). It outputs an allocation of items $A^\pi = [A_1, \ldots, A_n]$ where $v(A_i) \geq 1 - \epsilon$, for all $i \in [n]$.

$$\begin{aligned} & v(\mathsf{Small}) + \sum_{k=1}^{n} v(B_k) \geq n. \\ & v(B_k) \leq 1 \ \ \forall k \in [n]. \\ & |v_j| < \epsilon \ \ \forall j \in \mathsf{Small}. \end{aligned} \tag{4.16}$$

$$\begin{aligned} & v(\mathsf{Small}) + \sum_{k=1}^{n} v(B_k) \geq n(1 - \tfrac{\epsilon}{2}). \\ & v(B_k) \leq 1 - \tfrac{\epsilon}{2} \ \ \forall k \in [n]. \\ & |v_j| < \tfrac{\epsilon}{2} \ \ \forall j \in \mathsf{Small}. \end{aligned} \tag{4.17}$$

Algorithm 5 works as follows. It has $n - 1$ rounds. Each round starts with a bundle ('bag') from B^π. If the bag is valued at least $(1 - \epsilon)$, then it is assigned to some agent. If not, we first add all the unallocated Small chores to this bag. Then one by one we add the unallocated goods from Small until it is valued at least $(1 - \epsilon)$, and assign to some agent. After all rounds are done, in the last step, all remaining items from Small are added to the bag B_n. The next lemma proves the correctness of the algorithm.

Algorithm 5: Bag-filling to find $(1 - \epsilon)$-MMS allocation of identical agents

Input : $(n, \mathcal{M}, v)$, Partition of Big $\subseteq \mathcal{M}$: $B^\pi = [B_1, B_2, \ldots, B_n]$. Input satisfies Condition set (4.16) or (4.17)

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

$A^\pi \leftarrow \emptyset$, Small $\leftarrow \mathcal{M} \backslash$ Big
for $k \in \{1, \ldots, n-1\}$ **do**
  while $v(B_k) < 1 - \epsilon$ **do**
    $j \leftarrow \operatorname{argmin}_{j \in \mathsf{Small}} v_j$
    $B_k \leftarrow B_k \cup \{j\}$, Small $\leftarrow$ Small $\setminus \{j\}$
  $A^\pi \leftarrow A^\pi \cup \{B_k\}$
$B_n \leftarrow B_n \cup$ Small, $A^\pi \leftarrow A^\pi \cup \{B_n\}$
return A

Lemma 4.4.3. *If an* MMS *instance with identical agents satisfies condition set* (4.16) *or* (4.17)*, then Algorithm 5 gives a* $(1 - \epsilon)$*-Allocation.*

Proof. By induction on $k \in \{0, 1, 2, \cdots, n-1\}$, we prove that the value of each assigned bundle after k rounds is in $[1 - \epsilon, 1]$ if the instance satisfies condition set (4.16), and in $[1 - \epsilon, 1 - \epsilon/2]$ if it satisfies condition set (4.17). The base case when $k = 0$ is trivial.

First consider the case when condition set (4.16) is satisfied. Assume the value of all bundles assigned to the first $k-1$ agents are in this range. Now $v(B_j) \leq 1$ for all $j \in \{k, .., n\}$. If $v(B_k) \geq 1-\epsilon$, we are done. If not, then while the value of B_k is less than $1-\epsilon$, the value of the unallocated items from Small is at least the value of all items minus that of all the allocated bundles and unallocated bags of Big items. This can be bounded as,

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

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

Now suppose the instance satisfies condition set (4.17). In every round, while this is not true, the value of unallocated goods is at least $\epsilon/2$. Thus, there is at least one Small good. Finally, after assigning $n-1$ bundles, the total value remaining is at least $n(1-\epsilon/2) - (n-1)(1-\epsilon/2)$, hence the last bundle also has value at least $(1-\epsilon)$. QED.

4.4.4 The PTAS

We use the notions from the previous subsections and derive the PTAS, shown in Algorithm 6. The PTAS works as follows. It first enumerates all the partitions of Big. For each partition B^π, it first classifies the bundles into the $\mathcal{B}$-sets as per equation (4.15). If $\mathcal{B}_1$ is not empty, then add items from Small^- to any bag in $\mathcal{B}_1$, re-defining the sets and removing the assigned item from Small^- after each assignment. This process ends when either $\mathcal{B}_1 = \emptyset$ or $\mathsf{Small}^- = \emptyset$. In the first case, condition set (4.16) of the Bag-Fill algorithm is satisfied, and we run Algorithm 5 (Line 8) and return its output.

Otherwise when $\mathcal{B}_1 \neq \emptyset$, if $|\mathcal{B}_4| = 0$, then reduce to the following goods manna α-MMS problem instance $(\mathcal{N}', \mathcal{M}', v')$. $\mathcal{N}'$ is the set of agents who received bundles from $\mathcal{B}_3$ or $\mathcal{B}_4$. $\mathcal{M}'$ has (a) Small^+, with each item having the same value in v' as in v, and (b) for each bundle $B \in \mathcal{B}_3$, $\mathcal{M}'$ has a new item b with value $v'_b = v(B)$. Run the PTAS from [108] on $(\mathcal{N}', \mathcal{M}', v')$ to find a $(1-\epsilon)$-MMS allocation of $\mathcal{M}'$ among the $n' = n - (|\mathcal{B}_1| + |\mathcal{B}_2|)$ agents, and store its output in $\mathbb{A}$.

For the final case when $\mathcal{B}_1 \neq \emptyset$ and $|\mathcal{B}_4| > 0$, first check if the remaining unallocated goods and agents in $\mathcal{B}_3 \cup \mathcal{B}_4$ fulfill the condition set 4.17. If they do, apply the Bag-Fill and return the $(1-\epsilon)$-MMS allocation. If not, then B^π is invalid, hence discarded. After enumerating all the partitions, the algorithm returns the best allocation from $\mathbb{A}$.

Algorithm 6: $(1-\epsilon)$-MMS allocation for identical agents with MMS ≥ 0

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

$\mathbb{A} \leftarrow \emptyset$. $\Pi_n(\text{Big}) \leftarrow$ all partitions of Big into n sets.
for $B^\pi \in \Pi_n(\text{Big})$ **do**
  Define $\mathcal{B}$-sets as per equation (4.15).
  while $\mathcal{B}_1 \neq \emptyset$ *and* $\text{Small}^- \neq \emptyset$ **do**
    Remove any item j from Small^- and assign to any agent with a $\mathcal{B}_1$ bundle
    Re-define the $\mathcal{B}$-sets for the new allocation
  if $\mathcal{B}_1 = \emptyset$ **then**
    $A^\pi \leftarrow$ Bag-Fill($(\mathcal{N}, \mathcal{M}, v)$, $\mathcal{B}$-sets)
    return A^π
  $\mathcal{A}^{1,2} \leftarrow$ allocation of all bundles from $\mathcal{B}_1$ and $\mathcal{B}_2$ to distinct agents
  $\mathcal{N}' \leftarrow$ set of remaining agents, $\mathcal{M}' \leftarrow \cup_{B \in \mathcal{B}_3 \cup \mathcal{B}_4} B \cup \text{Small}^+$, $n' = |\mathcal{N}'|$
  if $\mathcal{B}_4 \neq \emptyset$ **then**
    if $v(\mathcal{M}') \geq n'(1 - \frac{\epsilon}{2})$ **then**
      $A^\pi \leftarrow \mathcal{A}^{1,2} \cup$ Bag-Fill($(\mathcal{N}', \mathcal{M}', v), \mathcal{B} - sets = \mathcal{B}_3 \cup \mathcal{B}_4$)
      return A^π
    else
      continue `// B^π is invalid`
  else
    $\mathcal{M}' \leftarrow \text{Small}^+$
    for $B \in \mathcal{B}_3$ **do**
      introduce a new good b with $v(b) = v(B)$; $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{b\}$
    $A^\pi \leftarrow \mathcal{A}^{1,2} \cup (1-\epsilon)$-MMS allocation for $(\mathcal{N}', \mathcal{M}', v')$ using the algorithm in [108]
    $\mathbb{A} \leftarrow \mathbb{A} \cup \{A^\pi\}$
return $\text{argmax}_{A^\pi \in \mathbb{A}} \min_{A_i \in A^\pi} v(A_i)$

Let us now discuss the analysis of the PTAS. To prove correctness when $\mathcal{B}_1 \neq \emptyset$ and $\mathcal{B}_4 = \emptyset$, we first show in Lemma 4.4.4 a relation between the MMS values of the given instance and the reduced goods manna instance. Let $A^{\pi *}$ be some MMS allocation, and $B^{\pi *} = \{B_1^*, B_2^*, \cdots, B_n^*\}$ be the allocation of Big items according to $A^{\pi *}$.

Lemma 4.4.4. *If for $B^{\pi *}$, the subsequent allocation of $\mathsf{Big} \cup \mathsf{Small}^-$ in Algorithm 6 has $\mathcal{B}_4 = \emptyset$, then,*

$$\mathsf{MMS}^n(\mathcal{M}) \leq_p \mathsf{MMS}^{n-|\mathcal{B}_1|-|\mathcal{B}_2|}(\bigcup_{B \in \mathcal{B}_3} B \cup \mathsf{Small}^+) \,.$$

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

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

Theorem 4.4.1. *Given an instance $(n, \mathcal{M}, v)$ with $\mathsf{MMS} \geq 0$, Algorithm 6 returns a $(1-\epsilon)$-MMS allocation in $O(m)$ time.*

Proof. First we prove the correctness of the algorithm. Note that no valid partition is discarded, as the procedure before deciding to discard a partition is exactly the procedure to determine if the partition is invalid. Consider a valid partition $B^{\pi *}$ corresponding to an MMS allocation $A^{\pi *}$, and its $\mathcal{B}$-sets as per (4.15). From Lemma 4.4.2, such a partition exists. After executing the while loop on Line 4, as every Small chore has absolute value at most $\epsilon/2$, upon adding the last chore before the value falls below 1, the value of every bundle to which a chore was added is still at least $1 - \epsilon/2$. After this, one of the cases based on which conditions from $\mathcal{B}_1 = \emptyset$ and $\mathcal{B}_4 = \emptyset$ are true gets executed. In every case, there is some allocation generated, as the partition is valid.

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

If the PTAS of [108] is called, then first, the agents receiving bundles from $\mathcal{B}_1$, $\mathcal{B}_2$, by definition of these sets, have value at least $1 - \epsilon \geq (1-\epsilon)$-MMS for their bundle. Also, as $B^{\pi *}$ corresponds to an MMS allocation, the MMS value for allocating the remaining items among the remaining agents, from Lemma 4.4.4, is at least the original MMS value. Hence, a $(1-\epsilon)$-MMS allocation of

the goods manna instance, combined with the allocations to the agents with the $\mathcal{B}_1$ and $\mathcal{B}_2$ bundles, is $(1-\epsilon)$-MMS.

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

For running time, note that every iteration of the for loop first allocates all Small chores, then either runs a bag-filling algorithm which takes $O(m)$ time, discards the iteration, or runs the PTAS of [108] which takes $O(2^{\tilde{O}(1/\epsilon)}n\log m) = o(2^{(1/\epsilon^2)}n\log m)$ time. In the worst case, every iteration takes $O(m + O(2^{1/\epsilon^2}n\log m))$ time. The for loop runs for $n^{|\mathsf{BIG}|}$ iterations, which from Lemma 4.4.1 is $O(n^{n/\tau\epsilon}) = 2^{O(n\log n/\tau\epsilon)}$. Hence, the total run time of the algorithm is $O(2^{n\log n/\tau\epsilon}(2^{1/\epsilon^2}n\log m + m)) = O(m)$ time. QED.

4.5 COMPUTING MMS WHEN MMS < 0

In this section we introduce the algorithm that finds a $(1-\epsilon)$-MMS allocation of an agent with MMS < 0 for an instance $(n, \mathcal{M}, v)$ and a constant $\epsilon > 0$, or equivalently, a $(1-\epsilon)$-MMS allocation of $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$ when there are identical agents with valuation function v (Algorithm 7). From Lemmas 4.1.2 and the normalization $v(\mathcal{M}) = -n$, we have MMS ≤ -1.

From Definition 4.1.1, a $(1-\epsilon)$-MMS allocation gives each agent a bundle with value at least $(1/(1-\epsilon))$MMS. Let $\sigma := \frac{1}{1-\epsilon} - 1$. Algorithm 7 obtains an allocation where each agent gets a bundle of value at least $(1+\sigma)$MMS $= (1/(1-\epsilon))$MMS. The high level idea of the algorithm is as follows. First we scale the valuations so that $v(\mathcal{M}) = -n$, and classify items as Big or Small. Then similarly as in Algorithm 6, we enumerate all partitions of Big. While there are unallocated Small goods, we add them one by one to the bundle with the least value. Once all the Small goods are exhausted, we iteratively add Small chores to the bundle with the highest value.

Theorem 4.5.1. *Algorithm 7 gives a $(1-\epsilon)$-MMS allocation when MMS < 0 in $O(m)$ time.*

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

$$v(\mathsf{Small}^+) \geq \mathsf{MMS}\cdot|\mathcal{S}| - v(B^{\pi*}) \tag{4.18}$$

Now, let $A^\pi = \{A_1, \ldots, A_n\}$ be the output of Algorithm 7. Suppose for contradiction there exists some $A_i \in A$ such that $v(A_i) < (\frac{1}{1-\epsilon})$MMS $= (1+\sigma)$MMS. Consider each $A_k \supseteq B_k$ with $B_k \in \mathcal{S}$. Note that the algorithm adds Small goods to the bundle with the least value. Because of A_i, before

Algorithm 7: $(1-\epsilon)$-MMS Allocation for identical agents with MMS < 0

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

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

adding the last Small good to any bundle, its value is less than $(1+\sigma)$MMS. The last good added has value at most σ. Therefore, all the A_ks have value at most MMS. From, (4.18) and the fact that the algorithm adds goods to the least valued bundle, we have,

$$v(\text{Small}^+ \setminus (\bigcup_{k \in [n]} A_k)) \geq \sum_{B \in \mathcal{S}} \text{MMS} - v(B) + \sigma = \sigma, \tag{4.19}$$

which is a contradiction.

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

Finally, $|\text{BIG}| = O(n/\sigma) = O(n/\epsilon)$, from the definition of BIG and σ. As every iteration corresponding to a partition of BIG takes $O(m)$ time, Algorithm 7 runs for $O(m \cdot 2^{O(n \log n/\epsilon)}) = O(m)$ time. QED.

4.6 HARDNESS OF APPROXIMATION

The α-MMS + PO problem makes two assumptions. First, the number of agents is assumed to be a constant. Second, the sum of absolute values of all the items for every agent is assumed to be at least τ times the minimum of this sum for the goods and the chores, for some constant $\tau > 0$.

In this section we show that relaxing either of these two assumptions makes the α-MMS problem NP-hard for any $\alpha \in (0, 1]$, even when agents are identical.

When agents are identical, the allocation that decides the MMS value of the agents is also an MMS allocation for the instance. Thus, for $\alpha = 1$, the GENERAL α-MMS problem should return an MMS allocation. Furthermore, given $v(\mathcal{M}) > 0$, we are guaranteed to have MMS ≥ 0 due to Lemma 4.1.1. However next we show that when either assumption of problem α-MMS is dropped, *deciding* if the inequality is indeed strict is NP-hard.

We separate Theorem 4.2.1 as two NP-hardness results in Theorems 4.6.1 and 4.6.2. To prove both, we reduce from the known NP-hard PARTITION problem.

PARTITION **Problem.** Given a set of non-negative integers $E = \{e_1, \ldots, e_m\}$, output YES if there exists a division of the elements into two sets of equal weight, otherwise output NO.

Theorem 4.6.1. *Given an instance $(n, \mathcal{M}, v)$ with constantly many (two) identical agents and $v(\mathcal{M}) > 0$, checking if* MMS > 0 *is* NP*-hard.*

Proof. We reduce an instance of PARTITION to an MMS instance $(n, \mathcal{M}, v)$ with two identical agents. Let $\mathcal{N} = \{1, 2\}$. $\mathcal{M} = [m + 2]$, where the first m items are goods and the last two are chores. The valuation function v is defined as follows, where $\beta = 1/4$.

$$v_j = \begin{cases} e_k, & \forall\, j \in [m] \\ -(\sum_k e_k/2) + \beta, & j \in \{m+1, m+2\}. \end{cases}$$

That is, the goods correspond to PARTITION elements, and have the same value as the weight of the element, and the chores are β more than the negated weight of each set in an equal distribution of the elements. Note that, (a) the trivial partition where all items are in the same bundle has the smaller bundle valued zero, and (b) the average of values of all items is β, and MMS cannot be higher than the average (Lemma 4.1.2). Hence, $0 \leq$ MMS $\leq \beta$.

We prove the correctness of the reduction in the following two claims.

Claim 4.6.1. PARTITION *has a solution* $\Rightarrow$ MMS $\geq \beta$.

Proof. Divide the goods into two bundles as per the PARTITION solution, and add one chore to each set. This gives us two bundles of equal value β, implying that MMS $\geq \beta$. QED.

Claim 4.6.2. MMS $> 0 \Rightarrow$ PARTITION *has a solution.*

Proof. We prove the contrapositive by contradiction. Suppose PARTITION does not have a solution. but MMS > 0 for the instance $(n, \mathcal{M}, v)$. Let $A^\pi = (A_1, A_2)$ be the allocation achieving the MMS value, and let $u_1 = v(A_1)$ and $u_2 = v(A_2)$. Then we have $u_1, u_2 > 0$.

First we prove that both the chores cannot be in the same bundle. If they are, and if all goods are not in this bundle, then the value of the bundle with chores is at most the sum of all except the smallest good. This is $(-\sum_k e_k + 2\beta) + (\sum_k e_k - \min_k e_k) \leq 1/2 - 1 < 0$. If every good and chore is in the same bundle, the value of the other bundle is 0. But as both u_1, u_2 are greater than 0, hence the chores are in separate bundles.

But then the value of the goods in each bundle is at least the total value minus the chore's value, i.e., for $i = 1, 2$, $v(A_i \cap \mathcal{M}^+) = u_i - (-\frac{1}{2}\sum_{k\in[m]} e_k + \beta) \geq \mathsf{MMS} - \beta + \frac{1}{2}\sum_k e_k > \frac{1}{2}\sum_k e_k - \beta$. Since $\beta = 1/4$ while $v(A_i \cap \mathcal{M}^+)$ and $\frac{1}{2}\sum_k e_k$ are integers, it follows that $v(A_i \cap \mathcal{M}^+) \geq \frac{1}{2}\sum_k e_k$. Then partition $(A_1 \cap \mathcal{M}^+, A_2 \cap \mathcal{M}^+)$ of $E = (e_1, \ldots, e_m)$ is a solution of the PARTITION problem, a contradiction. QED.

Claims 4.6.1 and 4.6.2 show that $\mathsf{MMS} > 0 \iff$ there is a solution to PARTITION.

When agents are identical, they agree on every item if it is a good or a chore, and therefore $\mathcal{M}^{gc} = \emptyset$. Therefore, v_i^+ and v_i^- as defined in Definition 4.1.2 are same as $v(\mathcal{M}^+)$ and $|v(\mathcal{M}^-)|$ respectively. QED.

Theorem 4.6.2. *Given a fixed constant $\tau > 0$, even if an instance $(n, \mathcal{M}, v)$ with identical agents satisfies $|v(\mathcal{M})| \geq \tau \cdot \min\{v(\mathcal{M}^+), |v(\mathcal{M}^-)|\}$, checking if $\mathsf{MMS} > 0$ is NP-hard.*

Proof. Again, we give a reduction from PARTITION. Let $E = \{e_1, e_2, \cdots, e_m\}$ be the set of elements given as input for PARTITION. Create an instance $(n, \mathcal{M}, v)$ as follows: $\mathcal{N}$ has n agents, where n will be fixed later based on the value of τ. $\mathcal{M} = \{1, 2, \cdots, m+n\}$ where the first $m + (n-2)$ items are goods, and the last 2 are chores. The valuation function v is defined as follows, where $\beta = 1/4$.

$$v_j = \begin{cases} e_j & \forall j \in [m] \\ \beta & \text{for } j \in \{m+1, ..., m+(n-2)\} \\ -\sum_{k\in[m]} e_k/2 + \beta & \text{for } j \in \{m+n-1, m+n\}. \end{cases}$$

That is, the first m goods have values equal to the weights of the corresponding elements of PARTITION. The remaining $(n-2)$ goods have value β each, and both the chores have value $-(\sum_i e_i/2) + \beta$. Fix n to satisfy $|v(\mathcal{M})| \geq \tau \cdot \min\{v(\mathcal{M}^+), |v(\mathcal{M}^-)|\}$, or equivalently $v(\mathcal{M}^+) \geq (1+\tau)|v(\mathcal{M}^-)|$, that is, $((n-2)\beta + \sum_i e_i) \geq (1+\tau)(\sum_i e_i + 2\beta)$.

We again have $0 \leq \mathsf{MMS} \leq \beta$. The lower bound because $v(\mathcal{M}) > 0$ and Lemma 4.1.1, and the upper bound because the average $v(\mathcal{M})/n$ is β and Lemma 4.1.2. The correctness is argued in the next two claims.

Claim 4.6.3. PARTITION *has a solution* $\Rightarrow \mathsf{MMS} \geq \beta$.

Proof. Divide the first m goods as per the division of the elements of PARTITION into equal valued sets, and add one chore to each bundle. From the remaining goods $\{m+1, ..., m+(n-2)\}$ give one each to the remaining $(n-2)$ bundles. The value of every bundle created is β. Hence, MMS $\geq \beta$. QED.

Claim 4.6.4. MMS $> 0 \Rightarrow$ PARTITION *has a solution.*

Proof. We prove the contrapositive of the statement, by contradiction. Suppose PARTITION instance $E = e_1, \ldots, e_m$ does not have a solution, but MMS > 0 for $(n, \mathcal{M}, v)$.

Given that there are exactly two chores, at least $(n-2)$ bundles have only goods and has to have at least one good. Furthermore, since e_is are positive integers and $\beta = 1/4$, each of these $(n-2)$ bundles have value at least β. Now, β being the upper bound on the MMS value, wlog we can assume that these $(n-2)$ bundles have exactly one good of the minimum value, namely β. This exhaust the goods $\{m+1, \ldots, m+(n-2)\}$ with value β. Therefore, the two chores and all goods corresponding to the PARTITION problem elements, and no other good, are in the remaining two bundles. Let these be the first two bundles A_1 and A_2.

Now by the same argument as in the proof of Claim 4.6.2, we can show that both A_1 and A_2 have positive value only if each contains exactly one chore and the total value of goods in each, namely $v(A_i \cap E)$ for $i = 1, 2$, is at least $\frac{1}{2}\sum_{i\in[m]} e_i$. Thus, $(A_1 \cap E, A_2 \cap E)$ is a solution to the PARTITION instance E, a contradiction. QED.

Claims 4.6.3 and 4.6.4 show that MMS > 0 for $(n, \mathcal{M}, v) \iff$ there is a solution to PARTITION. QED.

Theorems 4.6.1 and 4.6.2 show that even if we know that MMS ≥ 0 checking if it is strictly positive is NP-hard. Since for $\alpha \in (0, 1]$, MMS $> 0 \Leftrightarrow \alpha$MMS > 0, this essentially means, we can not find an α-MMS allocation for *any* value of $\alpha \in (0, 1]$ if either of the two conditions in α-MMS problem is dropped. The next theorem formalizes this.

Theorem 4.2.1. *For any instance $(n, \mathcal{M}, v)$ with identical agents and $v(\mathcal{M}) > 0$ such that exactly one of the following two holds: (a) either $n = 2$ or (b) $|v(\mathcal{M})| \geq \tau \cdot \min\{v(\mathcal{M}^+), |v(\mathcal{M}^-)|\}$ for a constant τ, finding an α-MMS allocation of $(n, \mathcal{M}, v)$ for any $\alpha \in (0, 1]$ is NP-hard.*

Even though an instance with identical agents is guaranteed to have an allocation where every agent gets at least the MMS value, i.e., 1-MMS allocation exists, Theorem 4.2.1 ruling out an efficient algorithm for finding α-MMS allocation any $\alpha \in (0, 1]$ is very striking. In light of this result, it is evident that even getting a PTAS, in other words finding $(1-\epsilon)$-MMS allocation, in case of identical agents is non-trivial and important.

CHAPTER 5: MAXIMIN SHARE WITH OXS VALUATIONS

In this chapter, we initiate the study of computing MMS allocations in the setting where the agents are assumed to have OXS valuation functions.

OXS functions have not been studied in the fair division literature, but hold particular importance in economic theory, for instance [104, 109, 110, 111, 112, 113]. Typically, in much of economics and game theory, the valuations of agents are assumed to be *beyond-additive*, and have a decreasing marginal returns property; this is essentially a *complement-free-kind* property, where the value of a set of resources is less than the sum of the values of any subsets that cover this set. [104] described a hierarchy of five valuation function classes that have this property. As is aptly said in [114], 'their work has set the tone for the study of valuation functions in Algorithmic Game Theory'. Furthermore, OXS functions have a rich structure, and can be syntactically defined in various ways: for instance, as depth 2 trees of ORs of XORs of additive functions [104], or using bipartite matchings [114], or matrices [115]. We use the definition via bipartite matchings, which intuitively is as follows. Every agent is associated with a weighted bipartite graph where all the goods form one side or part of vertices. The value of a set of goods for the agent is the value of the maximum weight matching in the graph induced by this set (see Figure 5.1 for an example).

The best known algorithmic and non-existence results for the MMS problem under OXS valuations are due to the results for submodular and additive cases, namely PTAS to find $1/3$-MMS allocation due to submodular and non-existence of $39/40$-MMS allocation due to additive. We note that, even when the number of agents is two, no better than a $1/3$-MMS algorithm is known. Given the strong connection of OXS functions with bipartite matching, and the richness of the mathematical structure this allows, it is natural to ask the following questions.

Q: For an efficient computation, can the barrier of $1/3$ *be broken for* OXS *functions?*

Q: Can the $39/40$ *factor non-existence result be improved when valuations are beyond-additive?*

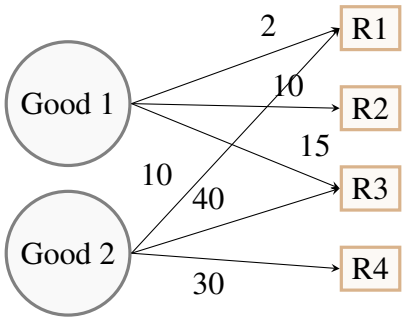


Figure 5.1: Example of a graph representation of an OXS function over 2 goods. The edges not shown have weight 0. The value of any set is the maximum weight matching value. $v(\{1,2\}) = 10 + 40 = 50$, $v(\{1\}) = 15$, and $v(\{2\}) = 40$.

We note that a negative result for OXS would extend to its super classes as well. Similarly, consider the task of computing the MMS value of any agent. This problem is equivalent to finding a 1-MMS allocation when the agents are identical. But solving this problem, even approximately up to any constant factor better than $1/3$ remains open for valuations that are beyond additive. This is in sharp contrast to the additive valuations where a PTAS is known [45], motivating the third question,

Q: Does a PTAS *exist for computing the* MMS *value for* OXS *valuations? If not, is there an algorithm for a factor better than* $1/3$*?*

We analyze all the three questions in this chapter, and as a result provide efficient algorithms, non-existence result, as well as hardness results, summarised below. In addition, we extend our algorithm to provide additional guarantees of EF1, PO, and *max social welfare* in special cases. This resolves existence of EF1 +PO allocation in this special case.

Results

Efficient algorithm. We show the existence of $\frac{1}{3}(1 + \frac{2/3}{(n-2/3)})$-MMS allocation, breaking the barrier of $1/3$ for the OXS valuations, and design a PTAS to compute one. As a corollary, it ensures much better factors for small number of agents, *e.g.,* $1/2$ with 2 agents, $3/7$ with 3 agents, $2/5$ with 4 agents and so on. To break the barrier of $1/3$ we uncover important properties of the OXS functions (w.r.t. the MMS problem). Importantly, we show that, every agent can assign one *representative value* to every good and as a result there is an ordering of the goods. Using this, we analyze the round-robin procedure to obtain a factor better than $1/3$. The challenge in the analysis is to bound the loss in value when a good is matched to an agent, as $O(n)$ goods get discarded due to this matching, due to being connected to the same right side vertex as the matched good in the OXS graph. These insights, together with a simple algorithm for beyond-additive valuations, may be of independent interest to analyze other fairness notions for OXS and other special classes of submodular functions, like gross-substitutes.

Non-existence of better than $2/3$**-MMS.** We show a simple example with 2 agents and 4 goods where an allocation strictly better than $2/3$-MMS does not exist. This gives an improved non-existence result for valuations that subsume OXS, namely *gross-substitutes*, *Rado*, and *submodular* valuations, as the previous best known result was for the submodular functions [50], with a non-existence of $3/4$-MMS allocations.

Computing MMS **value.** We show that the problem of computing MMS values of agents with OXS functions is strongly NP-hard, negating the possibility of a PTAS. To counter the impossibility result, we show an efficient algorithm to compute the MMS value of an agent within a factor of $1/2$.

EF1 +PO and EF1+MSW allocations with identical agents. A key subroutine of our algorithm with identical agents resolves another popular fair and efficient notion, namely the EF1+PO allocation. An allocation is called EF1 if every agent values their bundle more than any other agent's bundle upon removing some good from the other bundle. An allocation is called PO if there is no other allocation where every agent receives a bundle of equal or higher value, and at least one agent gets a strictly better bundle. Finding an EF1+PO allocation is a widely studied problem, with little success (the end of this section has a brief overview), and the existence of such an allocation is open, even with identical agents in the beyond-additive valuations setting.

We show such an allocation exists in the OXS valuations setting with identical agents. In fact, we show the existence of an allocation that is not only PO but also has the maximum social welfare (MSW), meaning the sum of values of all the bundles is maximized.

Challenges when showing MMS guarantees under OXS

Before discussing the details of our results, we discuss the key challenges that occur when trying to prove MMS guarantees under the OXS valuations setting. In this section, we follow the notation described in Section 5.1.

The first key challenge is to find an effective upper bound on the MMS values of an agent with an OXS function. Finding the MMS value of an agent is an NP-hard problem. Hence, in several works that prove α-MMS guarantees, first an upper bound on the MMS value is shown, that can be efficiently computed. The analysis then shows that their algorithm assigns to every agent a set valued at least a constant factor of this upper bound. Hence, for an upper bound to be effective, it must be constant factor away from the actual MMS value in *all* instances.

With additive valuation functions, a natural upper bound is the proportional value of all goods, that is, $v(\mathcal{M})/n$. With OXS valuations, the MMS value can be as high as $v(\mathcal{M})$ (for instance when there is a single right side vertex in the OXS graph, and every edge weight is the same) and as low as $v(\mathcal{M})/n$ (for instance in the special case of additive functions). Hence, the upper bound cannot be in terms of $v(\mathcal{M})$.

Another approach is to prove a bound as a function of the edge weights of the OXS function graph. As the OXS value of any subset of goods corresponds to the weight of a matching, this function must use at most one value associated with every good. The given OXS graph can have multiple values associated with every good, which further could be very different from each other. Hence, deciding which value to associate to a good is another critical challenge.

Another challenge is to bound the gain in value in each round of the round robin phase of the algorithm. The value from a good depends on the value from the other goods allocated to the agent in all other rounds. Further, assigning a good may result in the agent not being able to choose any

future good, as no other good adds positive value to any matching. Hence, having an upper bound on the loss due to the choice in each round is a challenge.

We get around these difficulties by observing some structural properties of the OXS *graph corresponding to an* MMS*-defining allocation*. This allows us to associate a single value to each good, that simultaneously yields for any agent, (i) a ranking of all the goods, and (iii) an upper bound on their MMS value. This ranking allows us to establish a bound on the minimum gain from the current round and the maximum loss in future choices due to the current choice. Section 5.2 discusses these insights in detail.

5.1 PRELIMINARIES

Notation. Let $[k]$ denote the set $\{1, 2 \cdots, k\}$.

We consider the problem of allocating a set $\mathcal{M}$ of m indivisible goods among a set $\mathcal{N}$ of n agents in a *fair* manner, with the fairness notion of *maximin share* (MMS). Each agent $i \in \mathcal{N}$ has a valuation function $v_i : 2^{\mathcal{M}} \to \mathbb{R}_{\geq 0}$ over subsets of the goods. Agents are called *identical* if their v_is are the same function; in this case, the valuation function is denoted by v.

We denote an instance of fair division problem by $(\mathcal{N}, \mathcal{M}, (v_i)_{i \in \mathcal{N}})$ and one with identical agents by $(n, \mathcal{M}, v)$. The valuation functions of the agents are assumed to be OXS functions, defined as follows.

OXS functions and their succinct description. An OXS valuation function for a set $\mathcal{M}$ of m goods is a monotone non-decreasing, non-negative set function $v : \mathcal{S} \subseteq [m] \to \mathbb{R}_{\geq 0}$ that is succinctly represented by a complete weighted bipartite graph. One part of vertices of this graph, termed the *left* part, has m vertices, and each of these vertices corresponds to a distinct good in $\mathcal{M}$. There is no constraint on the other or the *right* part of vertices, but we will shortly see that we can assume there are polynomially many (in $|\mathcal{M}|$) of these. Figure 5.1 shows such a representation of an example OXS function over two goods.

We denote the graph corresponding to an OXS function v for a set of goods $\mathcal{M}$ by $\mathsf{G}_\mathsf{v}(\mathcal{M})$, or simply by G_v when the set $\mathcal{M}$ is evident. We abuse notation slightly and refer to each vertex in the left part of G_v and its corresponding good by g. Next, we represent the sub-graph of $\mathsf{G}_\mathsf{v}(\mathcal{M})$ formed by restricting the left part to those corresponding to the goods in $\mathcal{S}$, that is by deleting the vertices corresponding to $\mathcal{M} \backslash \mathcal{S}$, by $\mathsf{G}_\mathsf{v}(\mathcal{S})$. Let $\mathsf{M}_\mathsf{v}(\mathcal{S})$ denote any *matching*, and $\mathsf{M}^*_\mathsf{v}(\mathcal{S})$ a *maximum weight matching*, of $\mathsf{G}_\mathsf{v}(\mathcal{S})$. Finally we let (g, r) denote the edge connecting a vertex g in the left part and r in the right part of G_v or its sub-graphs, and $w(g, r)$ its edge weight.

Definition 5.1.1 (OXS function value)**.** *Given the graph* $\mathsf{G}_\mathsf{v}(\mathcal{M})$*, the corresponding* OXS *function's value* $v(\mathcal{S})$ *for any subset of goods* $\mathcal{S} \subseteq \mathcal{M}$ *is defined as the value of the matching* $\mathsf{M}^*_\mathsf{v}(\mathcal{S})$.

That is,

$$v(\mathcal{S}) = \max_{\mathsf{M_v}(\mathcal{S})} \sum_{(g,r)\in \mathsf{M_v}(\mathcal{S})} w(g,r) = \sum_{(g,r)\in \mathsf{M_v^*}(\mathcal{S})} w(g,r).$$

Consider the set of top m edges (by weight) that are incident on a good, g. Since there are a total of m goods, we see that in no matching will a good be matched to an edge that does not belong to this set. Therefore, one can assume without loss of generality, that any vertex in the left part is adjacent to at most m edges. Consequently there are at most m^2 edges and hence at most m^2 vertices in the right part of any graph corresponding to an OXS function. We will assume hence forth the following property, and prove it in the Appendix for completeness.

Lemma 5.1.1. *The degree of any vertex in the left part of the graph corresponding to any* OXS *valuation function is at most m, where m is the number of vertices in the left part. Consequently, the number of vertices in the right part of such a graph are at most m^2.*

Proof. It suffices to show that by discarding all except the topmost m edges adjacent to any left vertex g in a graph corresponding to some OXS function, we do not affect the value of any subset $S \subseteq \mathcal{M}$ that contains g.

The value of g in the set S is equal to the weight of the edge adjacent to g in the maximum weight matching of S with the right part of vertices. Let the vertices in the right part of the graph that are connected to g via a non-zero edge weight be ordered in non-increasing order of their adjacent edge's weights. As there are m goods in all, $|S\backslash\{g\}| \leq m-1$. Therefore, in the worst case, at most the topmost $m-1$ right vertices according to the ordering determined above get matched to another good in S in its maximum weight matching, and the m^{th} highest right vertex can be matched to g. Consequently, the marginal value of g in the matching, and therefore in any set S, is at most equal to the m^{th} highest weight edge adjacent to g.

Finally, as there are m vertices in the left part, and each is connected to at most m right vertices, there are at most m^2 vertices in the right part. QED.

Next, let $A^\pi = \{A_1, A_2, \cdots, A_n\}$ denote a partition of all the goods among the n agents, referred as an *allocation*, i.e., $A_i \cap A_{i'} = \emptyset$ for all distinct i, i' in $\mathcal{N}$, and $\cup_i A_i = \mathcal{M}$. And let $\Pi_{\mathcal{N}}(\mathcal{M})$ be the set of all possible allocations of $\mathcal{M}$ among the agents in $\mathcal{N}$.

MSW. The social welfare (SW) of an allocation A^π is defined as the sum of values of all agents for their bundles of goods, that is, $\mathsf{SW}(A^\pi) = \sum_i v_i(A_i)$. The maximum social welfare (MSW) of an instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$ is the highest social welfare value over all allocations, that is,

$$\mathsf{MSW} = \max_{A^\pi \in \Pi_{\mathcal{N}}(\mathcal{M})} \sum_i v_i(A_i)$$

An allocation whose social welfare is equal to MSW is called an MSW allocation.

MMS values and α-MMS allocations. The maximin share (MMS) value of an agent i is defined as

$$\mathsf{MMS}_i^n(\mathcal{M}) = \max_{(A_1,\ldots,A_n)\in\Pi_{\mathcal{N}}(\mathcal{M})} \min_{k\in[n]} v_i(A_k).$$

We refer to $\mathsf{MMS}_i^n(\mathcal{M})$ by MMS_i when the qualifiers n and $\mathcal{M}$ are clear, and by MMS when the agents are identical. We will refer to the allocation that defines the MMS value of any agent i, that is, $\operatorname{argmax}_{\Pi_{\mathcal{N}}(\mathcal{M})} \min_{k\in[n]} v_i(A_k)$, as the MMS *defining allocation* of agent i. Note that MMS_i does not depend on the valuation functions of the other agents, and only depends on the total number of agents involved.

Definition 5.1.2 (α-MMS allocation)**.** *A^π is called an α-MMS allocation for an $\alpha \geq 0$, if for each agent $i \in \mathcal{N}$ we have $v_i(A_i) \geq \alpha\mathsf{MMS}_i$.*

Envy-free up to 1 good (EF1). An allocation A^π is called EF1 if for any two agents i, i', there is a good g in A_i such that $v_{i'}(A_{i'}) \geq v_{i'}(A_i\backslash\{g\})$. We will say i *is* EF1 *with respect to* i', if this inequality holds for some $g \in A_i$.

5.2 EXISTENCE OF $\frac{1}{3}(1 + \frac{2/3}{(n-2/3)})$-MMS ALLOCATION

In this section we will prove the following result.

Theorem 5.2.1. *There exists a $\frac{1}{3}(1 + \frac{2/3}{(n-2/3)})$-MMS allocation for every instance with OXS valuation functions.*

Our existence proof is constructive. The algorithm assumes we know the MMS value of every agent, hence currently not known to be efficient. We now describe the key ideas that form the building blocks of the algorithm. We will establish the existence of an α-MMS allocation for some variable α. In the end we optimize α to its highest possible value and show that this is $1/3(1 + 1/(n-1))$, thus proving Theorem 5.2.1.

5.2.1 Singleton set pre-assignments

It is well known that after assigning singleton sets of goods to any number of agents, the MMS value of an agent for the remaining goods when distributed among the remaining (number of) agents does not decrease [50]. Formally, $\mathsf{MMS}_i^{(n-|\mathcal{S}|)}(\mathcal{M}\backslash\mathcal{S}) \geq \mathsf{MMS}_i^n(\mathcal{M})$, for any agent i, number of agents $n \geq 1$, set of goods $\mathcal{M}$, and any subset $\mathcal{S} \subseteq \mathcal{M}$ with $|\mathcal{S}| \leq n$.

This allows us to perform the following *reduction*, on any fair division instance.

Theorem 5.2.2. *[Removing singleton goods] Given a fair division instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$, if there exists (i) an agent i and a good j such that $v_i(j) \geq \alpha\mathsf{MMS}_i$, and (ii) an α-MMS allocation A^{π}_{-i} of the reduced instance $(\mathcal{N}\backslash\{i\}, \mathcal{M}\backslash\{j\}, (v_k)_{k\in\mathcal{N}\backslash\{i\}})$, then the allocation $A^{\pi}_{-i} \cup \{A_i = \{j\}\}$ is an α-MMS allocation for the instance $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$.*

Theorem 5.2.2 can be used to design a first phase of the algorithm to eliminate all agent-good pairs (i, g) such that agent i has value at least α-MMS_i for any single good j. The next ideas are used to develop the second phase.

5.2.2 MMS defining graph of an agent

Given an MMS defining allocation of an agent, say $\mathcal{A}^{\mathsf{MMS}}$, and her valuation function, say v, create a bipartite graph, denoted by $\mathsf{G}^{\mathsf{MMS}}$, with all the goods on the left part, and the right part consisting of as many vertices as in the right part of G_{v}. For every bundle $\mathcal{A}_i^{\mathsf{MMS}}$ in $\mathcal{A}^{\mathsf{MMS}}$, consider the optimal matching $\mathsf{M}^*_{\mathsf{v}}(\mathcal{A}_{\mathsf{i}}^{\mathsf{MMS}})$. Add the edges from $\mathsf{M}^*_{\mathsf{v}}(\mathcal{A}_{\mathsf{i}}^{\mathsf{MMS}})$ to $\mathsf{G}^{\mathsf{MMS}}$ with the same edge weights.

We observe two properties of MMS defining graphs.

Lemma 5.2.1. *The degree of any vertex in the left part of an MMS defining graph is at most 1.*

Proof. Vertices in the left part correspond to some good, and the MMS defining graph corresponds to an allocation. Any good belongs in exactly one bundle in the allocation, and is matched to at most one vertex in the right part in the bundle's optimal matching. QED.

Lemma 5.2.2. *The degree of any vertex in the right part of an MMS defining graph is at most n.*

Proof. Every bundle's optimal matching can have at most one edge incident on a vertex in the right part of an MMS defining graph, and there are n bundles in the allocation. QED.

These graph properties will help in bounding the loss incurred by every agent in the second phase.

5.2.3 Ranking goods and upper bound on MMS

Using Lemma 5.2.1, we define the following ranking of all goods for every agent.

Definition 5.2.1 (Ranking of goods for every agent)**.** *Let the weight of a good be equal to the edge weight of the unique edge incident to the vertex corresponding to the good in the MMS-defining graph of the agent, or zero if the degree of this vertex is zero. Rank all the goods in the descending order of their weights, breaking ties arbitrarily.*

We denote the ranking of goods for an agent i by $\mathcal{R}_i$, the rank of a good j by r_{ij}, and the weight of the good j by w_{ij}.

This ranking, combined with Lemmas 5.2.1 and 5.2.2, helps in analyzing a minimum gain for every agent in Phase 2. To relate this gain to her MMS value, we prove the following upper bound on MMS_i based on $\mathcal{R}_i$.

Lemma 5.2.3. *[Upper bound on MMS] For every agent i, their MMS value is at most $1/n$ times the sum of weights of all the goods, that is, $\mathsf{MMS}_i \leq \sum_{j \in \mathcal{M}} w_{ij}/n$.*

Proof. Consider a bundle $\mathcal{A}_k^{\mathsf{MMS}}$ in an MMS allocation $\mathcal{A}^{\mathsf{MMS}}$ of agent i. The value of the bundle is the sum of the weights of the edges in the optimal matching of the bundle, that is, $\sum_{j:\exists(j,r)\in \mathsf{M}^*_{\mathsf{v}}(\mathcal{A}_k^{\mathsf{MMS}})} w_{ij}$. The sum of values of all the bundles of $\mathcal{A}^{\mathsf{MMS}}$ is

$$\sum_{k\in[n]} \sum_{j:\exists(j,r)\in \mathsf{M}^*_{\mathsf{v}}(\mathcal{A}_k^{\mathsf{MMS}})} w_{ij} = \sum_j w_{ij}.$$

As the MMS value of the agent is the smallest valued bundle's value in $\mathcal{A}^{\mathsf{MMS}}$, it is smaller than the average of all the bundle's values, thus proving the lemma. QED.

5.2.4 Putting it together

We now describe an algorithm that takes a fair division instance with OXS valuations, and the MMS values of all agents, and returns a $(1/3(1+1/(n-1)))$-MMS allocation.

Algorithm. The algorithm consists of two phases. First, while there is an agent i and a good j such that i values j to at least α-MMS_i, j is assigned to i, and both i and j are removed from the instance. In the second phase, we arrange all the agents arbitrarily. In a round robin fashion, every agent picks the good of the highest marginal value over their current allocation in each round.

Analysis of Algorithm 8. First, the following lemma for Phase 1 follows trivially from the condition checked while allocating the singleton good in this phase.

Lemma 5.2.4. *Any agent i who receives a bundle in Phase 1 of Algorithm 8 receives a bundle of value at least $\alpha\mathsf{MMS}_i$.*

Next, we set up some notation to analyze phase 2. We abuse notation slightly by referring to $\mathcal{R}_i$, MMS_i and $\mathsf{G}^{\mathsf{MMS}}$ as the ranking of goods, MMS value and the MMS defining graph of every agent i in the reduced instance obtained after Phase 1 of the algorithm. We also refer to $\mathcal{N}$, $\mathcal{M}$, n and m as the sets of agents and goods, and the number of agents and goods remaining after phase 1. Note that from Theorem 5.2.2, the MMS values of the agents do not decrease, hence proving that the agents receive α-MMS valued bundles according to the new MMS values suffices.

Algorithm 8: $(\frac{1}{3}(1+\frac{1}{(n\text{-}1)}))$-MMS allocation exists

Input : $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$, MMS_i for every $i \in \mathcal{N}$
Output: $(\frac{1}{3}(1+\frac{1}{(n\text{-}1)}))$-MMS allocation

Initialize $A^\pi \leftarrow (\emptyset)_{i\in\mathcal{N}}$, $\alpha \leftarrow \frac{1}{3}(1+\frac{1}{(n\text{-}1)})$
Phase 1**: Single goods pre-assignment**
while $\exists i \in \mathcal{N}, j \in \mathcal{M} :\ v_{ij} \geq \alpha\mathsf{MMS}_i$ **do**
    $A_i \leftarrow \{j\}, \mathcal{N} \leftarrow \mathcal{N}\backslash\{i\}, \mathcal{M} \leftarrow \mathcal{M}\backslash\{j\}$
Phase 2**: Round Robin**
Order all agents in $\mathcal{N}$ arbitrarily
while $\mathcal{M} \neq \emptyset$ **do**
    for $i \in [n]$ **do**
        if $\mathcal{M} \neq \emptyset$ **then**
            $j \leftarrow \text{argmax}_{k\in\mathcal{M}}\, v_i(A_i \cup \{k\}) - v_i(A_i)$ `// highest marginal valued good over current bundle`
            $A_i \leftarrow A_i \cup \{j\}, \mathcal{M} \leftarrow \mathcal{M}\backslash\{j\}$
return A^π

Let t^* denote the number of rounds and A_i^t denote the good obtained by agent i in round t of the round robin phase. We abuse notation slightly by referring to the weight of a good j of rank r by w_{ir} or simply w_r.

Lemma 5.2.5. *The marginal value of the good obtained by any agent i in any round t of the round robin phase of Algorithm 8 over their previous allocation is at least equal to the weight of the good of rank $(n-1)(2t-1)+1$. That is, $v_i(\cup_{k=1}^{t} A_i^k) - v_i(\cup_{k=1}^{t-1} A_i^k) \geq w_{(n-1)(2t-1)+1}$, for every $t \in [t^*]$.*

Proof. Consider an arbitrary round t. Agent i has already chosen $t-1$ goods in the previous rounds. Each of these goods will be incident to some vertex in the right part of their MMS defining graph. From Lemma 5.2.2, at most $n-1$ other goods are incident to the same vertex, making their marginal value over A_i^l zero. In the worst case, all the goods selected in the previous rounds are incident on distinct right side vertices, affecting the marginal value of some $(n-1)(t-1)$ goods. Further, in each round, agent i will be unable to pick goods of at most $n-1$ top most ranks from those still unallocated, because they may have gotten picked by the other agents in the same round before i's turn. Hence, in the worst case, i loses at most $(n-1)(t-1)$ goods to the other agents in the previous rounds, and a further $n-1$ goods in round t. Hence, the good of rank at most $(n-1)(t-1)+(n-1)(t-1)+(n-1)+1 = (n-1)(2t-1)+1$ has a marginal value over their current allocation equal to the weight of the good. QED.

Lemma 5.2.6. *The value of the bundle received by any agent i in the round robin phase of Algorithm 8 is at least $\frac{1}{2(n-1)}(n-(n-1)\alpha)\mathsf{MMS}_i$.*

Proof. From Lemma 5.2.5, we know that the value of every agent's allocation is at least,

$$v_i(A_i) \geq v_i(A_i^1) + \sum_{t=2}^{t^*}(v_i(\cup_{k=1}^{t} A_i^k) - v_i(\cup_{k=1}^{t-1} A_i^k)).$$

From Lemma 5.2.5, this is equal to the sum of weights of goods of ranks $(n-1)(2t-1)+1$, for all $t \in [t^*]$. That is,

$$v_i(A_i) \geq \sum_{t=1}^{t^*} w_{(n-1)(2t-1)+1}. \tag{5.1}$$

By definition of $\mathcal{R}_i$, we have $w_{ij} \geq w_{ij'}$ when $j \geq j'$. Thus,

$$w_{ij} \geq \frac{1}{l-j}\sum_{k=j}^{l} w_{ik},\ \forall j, l :\ j < l$$

Using $l = j + 2(n-1)$ and combining with (5.1), we get,

$$\begin{aligned} v_i(A_i) &\geq \sum_{t=1}^{t^*} \frac{1}{2(n-1)} \sum_{k=(n-1)(2t-1)+1}^{(n-1)(2t+1)} w_k \\ &\geq \frac{1}{2(n-1)} \sum_{j \in [m]\setminus[n-1]} w_j. \end{aligned} \tag{5.2}$$

Now as we have pre-assigned all goods that have value α-MMS_i or more in round 1, each of the first $n-1$ ranked goods has value at most α-MMS_i.

Also from Lemma 5.2.3, $\sum_j w_j. \geq n \cdot \mathsf{MMS}_i$. Hence, we can further simplify equation (5.2) to,

$$v_i(A_i) \geq \frac{1}{2(n-1)}(n-(n-1)\alpha)\mathsf{MMS}_i. \qquad \text{QED.}$$

Proof of Theorem 5.2.1. To ensure agents receive a bundle of at least α-MMS_i value in Phase 2, from Lemma 5.2.6, we require,

$$\frac{1}{2(n-1)}(n-(n-1)\alpha)\mathsf{MMS}_i \geq \alpha\mathsf{MMS}_i.$$

Simplifying this expression, we get,

$$\begin{aligned} \alpha &\leq \frac{1}{2(n-1)}(n-(n-1)\alpha) \\ \Rightarrow \alpha &\leq \frac{1}{3}(1 + \frac{2/3}{(n-2/3)}). \end{aligned}$$

Combined with Lemma 5.2.4, Theorem 5.2.1 follows.

5.3 PTAS FOR $\frac{1}{3}(1+\frac{2/3}{(n-2/3)})$-MMS ALLOCATION

Using the existence proof of a $\frac{1}{3}(1+\frac{2/3}{(n-2/3)})$-MMS allocation, we provide a PTAS for computing one.

Theorem 5.3.1. *There is a* PTAS *that given an instance* $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$ *and an* $\epsilon > 0$, *computes a* $(\frac{1}{3}(1+\frac{2/3}{(n-2/3)})-\epsilon)$-MMS *allocation in time* $O(mn^2 \log_{(1+\epsilon)} v(\mathcal{M}))$, *where* $v(\mathcal{M}) = \max_i v_i(\mathcal{M})$.

Note that Algorithm 8 would be an efficient algorithm if it did not require the knowledge of the MMS values of all agents. Informally, we get around this requirement by (1) guessing all the MMS values, referring to these guesses as the *threshold* values of the agents, and (2) running an exponential search to find the correct set of threshold values in polynomial time. We prove this can be done in polynomial time by showing that, finding the correct threshold of an agent, is a process independent of the other agents. We now discuss the details of the proof.

Algorithm. We define a value called the *threshold value* of agent i, and denote it by τ_i. Initialize each τ_i to $v_i(\mathcal{M})$, that is, the value of agent i for all the goods $\mathcal{M}$. Run Algorithm 1 by using each τ_i instead of MMS_i whenever required. If the allocation returned fails to assign some agent i a value at least $\alpha\tau_i$, reduce the threshold of i to $\tau_i/(1+\epsilon)$, and repeat.

Algorithm 9: $(\frac{1}{3}(1+\frac{2/3}{(n\text{-}2/3)}))$-MMS allocation PTAS

Input : $(\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}})$, $\epsilon > 0$
Output: $(\frac{1}{3}(1+\frac{1}{(n\text{-}1)})-\epsilon)$-MMS allocation

Initialize $\alpha \leftarrow \frac{1}{3}(1+\frac{2/3}{(n\text{-}2/3)})$, $\tau_i \leftarrow v_i(\mathcal{M})$ for all $i \in \mathcal{N}$
$A^\pi \leftarrow$ Algorithm 8 on $((\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}}), (\tau_i)_{i\in\mathcal{N}})$
while $\exists i \in \mathcal{N} : v_i(A_i) < \alpha\tau_i$ **do**
    $\tau_i \leftarrow \frac{1}{(1+\epsilon)}\tau_i$
    $A^\pi \leftarrow$ Algorithm 8$((\mathcal{N}, \mathcal{M}, (v_i)_{i\in\mathcal{N}}), (\tau_i)_{i\in\mathcal{N}})$
return A^π

Lemma 5.3.1. *For any* $i \in \mathcal{N}$, *if* $\tau_i \le \mathsf{MMS}_i$, *then Algorithm 9 assigns agent* i *a bundle of value at least* $\alpha\tau_i$.

Proof. First, if agent i gets a good from Phase 1, then the value of the good trivially is at least $\alpha\tau_i$. Otherwise, note that the ranking of the goods of agent i does not depend on MMS_i. Hence, Lemma 5.2.5 holds for i in every iteration of Algorithm 9, irrespective of the threshold value τ_i considered in that iteration. Similarly, equation (5.1) in the proof of Lemma 5.2.6 also holds for every iteration. When $\tau_i \le \mathsf{MMS}_i$, then $\sum_j w_j \ge n\mathsf{MMS}_i \ge n\tau_i$. Hence, the remaining analysis of the proof also goes through and Lemma 5.2.6 holds. Thus, when $\tau_i \le \mathsf{MMS}_i$, the analysis of Algorithm 8 holds, and i's value for her bundle is at least $\alpha\tau_i$. QED.

Theorem 5.3.1 follows immediately from the lemma as follows.

Proof of Theorem 5.3.1. Using Lemma 5.3.1, we know that if $\tau_i \geq \mathsf{MMS}_i$, then Algorithm 9 will allocate a bundle of value at least α-MMS_i to agent i. We also know that the MMS value of i cannot be more than $v_i(\mathcal{M})$, their value for all the goods, as it is the value of some subset of goods in an MMS defining allocation. Thus, if we start at the threshold value $\tau_i = v_i(\mathcal{M})$, and keep reducing the value of τ_i until i receives a bundle of value $\alpha\tau_i$, the value of τ_i at which this process ends is $\mathsf{MMS}_i/(1+\epsilon)$. This is because in the previous round, as we had to reduce the threshold value, this threshold, say τ_i^{prev}, was greater than MMS_i. The next threshold, say $\tau_i^* = \tau_i^{prev}/(1+\epsilon) > \mathsf{MMS}_i/(1+\epsilon)$. As we did not further reduce the threshold value, agent i received a bundle of value at least $\alpha\tau_i^* > \alpha\mathsf{MMS}_i/(1+\epsilon). > \alpha(1-\epsilon)\mathsf{MMS}_i \geq (\alpha-\epsilon)\mathsf{MMS}_i$.

Finally let us discuss the running time of the algorithm. First, note that every run of Algorithm 8 takes $O(mn)$ time due to Phase 1. This algorithm is called as a subroutine until every agent's threshold is set to its correct value, that is in $(\mathsf{MMS}_i/(1+\epsilon), \mathsf{MMS}_i]$. Now each agent's threshold value can take $O(\log_{(1+\epsilon)} v_i(\mathcal{M}))$ iterations to be correctly set. Thus, the number of times Algorithm 8 is called, hence the total time of Algorithm 9, is $O(mn \times \sum_i \log_{(1+\epsilon)} v_i(\mathcal{M})) = O(mn^2 \log_{(1+\epsilon)} v(\mathcal{M}))$, where $v(\mathcal{M}) = \max_i v_i(\mathcal{M})$.

5.4 IDENTICAL AGENTS

In this section, we establish the following result.

Theorem 5.4.1. *Given an instance $(n, \mathcal{M}, v)$ with identical agents and any $\epsilon > 0$, there exists a polynomial time algorithm that returns a $(\frac{1}{2} - \epsilon)$MMS allocation. The running time of the algorithm is $O(nm^3 \log_{(1+\epsilon)} v(\mathcal{M}))$.*

Overview. Broadly, the algorithm works as follows. We will guess the MMS value of the agent and run a subroutine, which returns a $1/2$-MMS allocation for the correct guess. We run an exponentially fast search over the possible MMS values to find the correct guess. In the subroutine, first, using Theorem 5.2.2, we allocate goods of value at least $1/2$-MMS as singleton sets. Now it is well known that an MSW allocation can be found for an instance with OXS valuations in polynomial time. For the remaining instance with no high valued goods, we start with such an allocation. We prove that while the allocation is MSW, and the value of the highest valued bundle, say v^{max}, is more than twice that of the lowest valued bundle, say v^{min}, we can always find a new allocation that maintains the social welfare and increases the value of v^{min} [Lemma 5.4.2]. We use this lemma repeatedly, and prove that the process converges in polynomial time in a $1/2$-MMS allocation. We discuss these ideas in detail.

MSW **allocation.** OXS valuation functions are a subclass of the well studied class of gross substitute valuations [104]. It is well known that an MSW allocation can be found in polynomial time for any instance where the agents have gross substitute valuations for the goods (see Section *Welfare problem for Gross Substitutes* in [116] for a survey reviewing multiple techniques to obtain such an allocation).

New MSW allocation that improves MMS guarantee. Lemma 5.4.2 is key to our iterative method that switches to other MSW allocations with improved MMS guarantees, converging in an allocation that is $1/2$-MMS.

Hence forth, we will call an allocation A^π *nice* if it is MSW and every good j has value $v(\{j\})$ at most $\mathsf{MMS}/2$. A nice allocation is called *improving*, if the smallest and largest valued bundles, say A_{min} and A_{max}, satisfy (a) $v(A_{max}) > 2v(A_{min})$ and (b) $v(A_{max}) \geq \mathsf{MMS}$. A nice allocation is called *improved*, if $v(A_i) \geq \mathsf{MMS}/2$ for all $i \in [n]$. We first show the following property, followed by the key lemma.

Lemma 5.4.1. *Every nice allocation is either improving or improved.*

Proof. Every nice allocation A^π is MSW, hence its social welfare is at least $n\mathsf{MMS}$. As the largest bundle's value is at least the average, $v(A_{max}) \geq \mathsf{MMS}$. Suppose A^π is not improving. Then by definition of improving allocations, $v(A_{max}) \leq 2v(A_{min})$. As $v(A_{max}) \geq v(A_i)$ for all i,

$$v(A_i)/v(A_{min}) \leq 2 \text{ for all } i \in [n]. \tag{5.3}$$

Adding all bundle values, and combining with the fact that the social welfare is at least $n\mathsf{MMS}$, we get,

$$\sum_i 2v(A_{min}) \geq n\mathsf{MMS} \Rightarrow v(A_{min}) \geq \frac{\mathsf{MMS}}{2}.$$

As $v(A_i) \geq v(A_{min})$ for all i, A^π is improved.

QED.

Given an allocation, A^π, for each bundle, $A_i \in A^\pi$, we can fix a maximum matching that achieves the value of this bundle, $\mathsf{M}^*_\mathsf{v}(\mathsf{A_i})$. In the next lemma, we describe a procedure to compute an improving allocation from another such that in both the allocations, all the goods are matched to the same right vertices, *even if they are moved to different bundles.*

Lemma 5.4.2. *Given an improving allocation A^π, an allocation $A^{\pi\prime}$ can be obtained in polynomial time that is nice and satisfies $v(A'_{min}) > v(A_{min})$, where A'_{min} is the bundle in $A^{\pi\prime}$ obtained after modifying A_{min}. Moreover, the goods are matched to same right hand side vertices in both A^π and $A^{\pi\prime}$.*

Proof. Consider the optimal matchings of the bundles $\mathsf{M}^*_{\mathsf{v}}(\mathsf{A}_{\mathsf{max}})$ and $\mathsf{M}^*_{\mathsf{v}}(\mathsf{A}_{\mathsf{min}})$ in the improving allocation A^π, referred respectively as $\mathsf{M}^*_{\mathsf{max}}$ and $\mathsf{M}^*_{\mathsf{min}}$ hence forth. Let there be k edges in $\mathsf{M}^*_{\mathsf{max}}$, denoted as e_j^{max} or $(g_j, r_j)^{max}$, for all $j \in [k]$, and l edges in $\mathsf{M}^*_{\mathsf{min}}$, denoted by $e_j^{min} = (g_j, r_j)^{min}$, $j \in [l]$. We assume without loss of generality that $w(e_j^a) \geq w(e_{j'}^a)$, for all $j \leq j'$ and $a \in \{min, max\}$, i.e., the edge weights are ranked in non-increasing order. We will also assume all edge weights are positive, as we can eliminate all the zero weight edges from both matchings without changing the properties of A^π.

We consider two cases. First, when there is a vertex r in the right part of $\mathsf{M}^*_{\mathsf{max}}$ that is unmatched in $\mathsf{M}^*_{\mathsf{min}}$. As A^π is an allocation, the good g that r is matched to does not belong in A_{min}. Thus, this edge $(g, r) \in \mathsf{M}^*_{\mathsf{max}}$ can be transferred to A_{min} resulting in the modified bundles $A'_{max} = A_{max} \backslash \{g\}$, and $A'_{min} = A_{min} \cup \{g\}$.

In the second case, suppose that all vertices in the right part of $\mathsf{M}^*_{\mathsf{max}}$ are matched in $\mathsf{M}^*_{\mathsf{min}}$. Consider the subgraph, say $\mathsf{M}^*_{\mathsf{min}}{}^{\mathsf{k}}$, of $\mathsf{M}^*_{\mathsf{min}}$ which has the k edges adjacent to these right part vertices, and the corresponding bundle, say A^k_{min}. We have $v(A_{max}) > v(A_{min}) \geq v(A^k_{min})$, i.e., the sum of the weights of the edges adjacent to these vertices in A_{max} is greater than this sum in A^k_{min}. Hence, there is at least one vertex r such that its adjacent edge $(g, r) \in A_{max}$ has higher value than the edge $(g', r) \in A^k_{min}$. We exchange these goods resulting in the bundles $A'_{max} = A_{max} \backslash \{g\} \cup \{g'\}$, and $A'_{min} = A_{min} \backslash \{g'\} \cup \{g\}$.

In the corresponding new allocation in all these cases, $A^{\pi\prime} = A^\pi \backslash \{A_{max}, A_{min}\} \cup \{A'_{max}, A'_{min}\}$, we have $v(A'_{min}) > v(A_{min})$. Further, in each bundle's optimal matching, all the goods can be connected to the same right part vertices as in A^π and the social welfare is maintained. Hence, the new allocation is also nice.

Finally, notice that in the new allocation, the goods can be matched to same vertices as original ones and this is still a feasible matching with each good having the same marginal contribution as in the previous matching. Suppose for one of the bundles, A_{min} or A_{max} we had to reassign the goods to new vertices. In this case, the new matching must give a strictly larger value than the old one. This implies that we would get a strictly higher social welfare with the new matchings giving a contradiction to the fact that we started with a nice allocation.

QED.

Algorithm. Algorithm 10 checks if MMS is zero, and returns a trivial allocation in this case. Otherwise, it runs an exponentially fast search for a correct guess of the MMS value. For each guess τ, it runs a subroutine, which has two phases. First, it allocates all goods of value $\tau/2$ or higher as singleton sets. Over the remaining agents and goods, it starts with an MSW allocation, and uses Lemma 5.4.2 to repeatedly modify it until the conditions of the lemma do not hold. We show that while τ is less than MMS, we get an allocation that gives each agent a bundle of value at least $\tau/2$. It returns the allocation obtained for the highest guess.

Algorithm 10: $(\frac{1}{2}-\epsilon)$-MMS allocation

Input : $(n, \mathcal{M}, v),\ \epsilon > 0$
Output: $(\frac{1}{2}-\epsilon)$-MMS allocation

Let $u \leftarrow \min_{S:|S|=n-1} \max_{j \in \mathcal{M}\setminus S} v(\{j\})$ // n^{th} good
if $u = 0$ // MMS is zero
then
  return $A^{\pi} = (\emptyset)^{(n-1)} \cup (\mathcal{M})$ // all goods to one agent
Initialize $\tau \leftarrow u,\ Flag \leftarrow 1, n^* \leftarrow n$
while $Flag$ **do**
  Phase 1**: Single goods pre-assignment**
  Let $i \leftarrow n$
  while $\exists j \in \mathcal{M} :\ v(\{j\}) \geq \frac{1}{2}\tau$ **do**
    $A'_i \leftarrow \{j\},\ i \leftarrow i-1, n \leftarrow n-1, \mathcal{M} \leftarrow \mathcal{M}\setminus\{j\}$
  Phase 2**: Improving** MMS **guarantee of** MSW
  Initialize $A^{\pi\prime} \leftarrow$ MSW allocation of $(n, \mathcal{M}, v)$ // use any method from survey [116]
  while $A^{\pi\prime}$ *is not improved* **do**
    $A^{\pi\prime} \leftarrow$ allocation using Lemma 5.4.2
  if $\exists\ A'_j$ *with* $v(A'_j) < \tau/2$ **then**
    $Flag \leftarrow 0$
    continue
  $A \leftarrow A',\quad \tau \leftarrow \tau \cdot (1+\varepsilon)$
return $A^{\pi} \cup (A_{i+1}, A_{i+2} \cdots, A_{n^*})$

The main components in the analysis of Algorithm 10 are the Lemmas 5.4.1 and 5.4.2. Let us discuss the remaining details.

Lemma 5.4.3. *If* $\tau \leq$ MMS, *the allocation returned by Algorithm 10 gives every agent a bundle of value at least* $\tau/2$.

Proof. We prove the lemma by dividing agents into two types: those who receive a bundle in phase 1, or in phase 2 of Algorithm 10. First, every agent who receives a bundle in the first phase of the algorithm trivially receives a good of value at least $\tau/2$.

For the second phase, we will abuse notation by denoting the instance $(n, \mathcal{M}, v)$ and its parameters as those of the reduced instance after phase 1. Note that as $\tau \leq$ MMS, from Theorem 5.2.2, the MMS value of the reduced instance, say MMS^r, is at least that of the original instance. Thus, every unallocated good after Phase 1 has value at most $\tau/2 \leq$ MMS$/2 \leq$ MMS$^r/2$, hence the MSW allocation A^{π} is nice. Consequently, the value of τ is not updated. From Lemma 5.4.2, by recursively applying it to an improving allocation, one always obtains a nice allocation. Each

application of Lemma 5.4.2 increases the value of the smallest bundle. Hence this process will converge with every agent receiving a bundle of value at least $\tau/2$. Combined with Phase 1, we obtain an allocation where every agent gets value at least $\tau/2$. QED.

Lemma 5.4.4. *Given an instance $(n, \mathcal{M}, v)$ and any $\epsilon > 0$, Algorithm 10 returns a $(1/2-\epsilon)$-MMS allocation.*

Proof. By arranging the goods in decreasing order of the value of the singleton set containing this good, we can create a trivial allocation where the first n goods are distributed in n distinct bundles, and the remaining goods allocated arbitrarily. One can see that $\mathsf{MMS} > 0$ if and only if in this allocation, the smallest bundle has non-zero value. If $\mathsf{MMS} = 0$, then any allocation is $1/2$-MMS, and returned. Otherwise, let u be the value of the n^{th} smallest good. We know trivially that $u \leq \mathsf{MMS} \leq v(\mathcal{M})$. Hence, initially, $\tau \leq \mathsf{MMS}$. As we increase τ to $\tau \cdot (1+\epsilon)$, in some iteration, $\tau \geq \mathsf{MMS}$. From Lemma 5.4.3, the Algorithm converges when this happens. Let τ^{next} be the value of τ when the algorithm converges, and τ^* be its value in the previous iteration. As $\tau^* < \mathsf{MMS}$, and $\tau^{next} = \tau^* \cdot (1+\epsilon)$, we have $\tau^* > \mathsf{MMS}/(1+\epsilon)$. From Lemma 5.4.3, every agent receives a bundle of value at least,

$$\frac{\tau^*}{2} > \frac{\mathsf{MMS}}{2(1+\epsilon)} \geq \frac{\mathsf{MMS}}{2}(1-\epsilon) \geq (\frac{1}{2} - \epsilon)\mathsf{MMS}.$$

QED.

Lemma 5.4.5. *Given an instance $(n, \mathcal{M}, v)$ and any $\epsilon > 0$, Algorithm 10 runs in $O(nm^3 \log_{(1+\epsilon)} v(\mathcal{M}))$ time.*

Proof. Let us analyze the time taken to complete Phase 2 once. Divide the agents into three groups, according to the value of their bundles at the beginning of any iteration of Phase 2 as follows. Let A_L^t be the set of all agents i for whom $v_i(A_i) < \frac{\mathsf{MMS}}{2}$ at the beginning of the t^{th} iteration, similarly let A_G^t be the agents i such that $v(A_i) > \mathsf{MMS}$, and A_M^t are the remaining agents, that is, $A_M^t := \mathcal{N} \setminus \{A_L^t \cup A_G^t\}$. By definition, the agent who participates with the smallest valued bundle in the t^{th} iteration belongs in A_L^t, while the one with the largest bundle is from A_G^t.

First note that an agent i in a set A_L^t for any t never belongs in any set $A_G^{t'}$, for any $t' \geq t$. This is because the value of i's bundle at the beginning of the t^{th} iteration is less than $\tau/2$, and any good has marginal value at most $\tau/2$. As the value of i's bundle increases only when it is chosen from some A_L^r, their bundle's value after any iteration where they were chosen cannot exceed beyond MMS, but to belong in a set $A_G^{t'}$ their value must be strictly higher than MMS. Therefore, any agent in any A_L^t (alternatively A_G^r) always belongs in $A_L^{t'} \cup A_M^{t'}$ for every t' higher than t (respectively r).

Now fix the agent $i \in A_L^1$ that is chosen as the agent with the smallest bundle. From Lemma 5.4.2, the algorithm will either add a good to i and match it to a new right vertex (not currently

matched to any good in i), or re-match an already matched right vertex to another good with a higher marginal value, that is a higher weighted edge adjacent to this right vertex. Thus, a pair (agent i, right vertex r) can participate in any iteration of Phase 2 as *the smallest bundle's agent i that matches a good to vertex r in the iteration*, at most m times. From Lemma 5.1.1, there are at most m^2 right hand vertices, hence an agent participates as the chosen agent from some A_L^t in at most m^3 iterations (for at most m^3 values of t). As there are n agents overall, the algorithm runs in $O(nm^3)$ iterations.

Finally, Phase 1 takes linear time. Thus, for each guess of MMS, the algorithm takes $O(nm^3)$ time. As we guess the value of MMS using powers of $(1+\epsilon)$, for any fixed constant $\epsilon > 0$, the algorithm runs in time $O(nm^3 \log_{(1+\epsilon)} v(\mathcal{M}))$. QED.

5.5 EF1+MSW ALLOCATION.

Theorem 5.5.1. *An* EF1+MSW *allocation exists for an instance $(n, \mathcal{M}, v)$ when all the agents have an* OXS *valuation function.*

Algorithm.

Algorithm 11: EF1+MSW allocation

Input : $(n, \mathcal{M}, v)$
Output: EF1+MSW allocation

Initialize $A^\pi \leftarrow$ MSW allocation of $(n, \mathcal{M}, v)$ `// use any method from survey [116]`
while A^π *is not* EF1 **do**
  Modify A^π by making the smallest and largest bundles EF1 with respect to each other `// use Lemma 5.5.1`
return A^π

Analysis.

We prove that recursively applying a method similar to that described in the proof of Lemma 5.4.2 to an MSW allocation results in an EF1+MSW allocation, thus proving the existence of such an allocation. The key component of the proof is Lemma 5.5.1. Its proof is similar to the proof of Lemma 5.4.2.

Lemma 5.5.1. *If an allocation A^π has two agents i, i' such that i' is not* EF1 *with respect to i, that is, for every good $g \in A_i$, $v(A_{i'}) < v(A_i \backslash \{g\})$, then one can obtain another allocation $A^{\pi'}$ where both i and i' are* EF1 *with respect to each other, and the social welfare is maintained, that is* $\mathsf{SW}(A^{\pi'}) \geq \mathsf{SW}(A^\pi)$.

Proof. Consider the optimal matchings of the bundles $\mathsf{M}^*_{\mathsf{v}}(\mathsf{A}_{\mathsf{i}})$ and $\mathsf{M}^*_{\mathsf{v}}(\mathsf{A}_{\mathsf{i}'})$ in the improving allocation A^π, referred respectively as $\mathsf{M}^*_{\mathsf{i}}$ and $\mathsf{M}^*_{\mathsf{i}'}$ hence forth. Let there be k edges in $\mathsf{M}^*_{\mathsf{i}}$, denoted as e^i_j or $(g_j, r_j)^i$, for all $j \in [k]$, and l edges in $\mathsf{M}^*_{\mathsf{i}'}$, denoted by $e^{i'}_j$ or $(g_j, r_j)^{i'}$, for all $j \in [l]$. We assume without loss of generality that $w(e^a_j) \geq w(e^a_{j'})$, for all $j \leq j'$ and $a \in \{i, i'\}$, that is, the edge weights are ranked in non-increasing order. We will also assume all edge weights are positive, as we can eliminate all the zero weight edges from both matchings without changing the properties of A^π.

We first consider two cases, (a) $l < k$, and (b) $l \geq k$, and there is a vertex r in the right part of $\mathsf{M}^*_{\mathsf{i}}$ that is unmatched in $\mathsf{M}^*_{\mathsf{i}'}$. In the former, there exists some edge $(g, r) \in \mathsf{M}^*_{\mathsf{i}}$ that is not in $\mathsf{M}^*_{\mathsf{i}'}$. In either case, we can transfer the edge (g, r) adjacent to r to $\mathsf{M}^*_{\mathsf{i}'}$, resulting in the modified bundles $A'_i = A_i \backslash \{g\}$ and $A'_{i'} = A_{i'} \cup \{g\}$.

The remaining case is when $l \geq k$ and every vertex r in the right part of $\mathsf{M}^*_{\mathsf{i}}$ is matched to some vertex in $\mathsf{M}^*_{\mathsf{i}'}$ as well. Consider the subgraph, say $\mathsf{M}^{*\mathsf{k}}_{\mathsf{i}'}$, of $\mathsf{M}^*_{\mathsf{i}'}$ which has the k edges adjacent to these right part vertices, and the corresponding bundle, say $A^k_{i'}$. We have $v(A_i) > v(A_{i'}) \geq v(A^k_{i'})$, that is, the sum of the weights of the edges adjacent to these vertices in A_i is greater than this sum in $A^k_{i'}$. Hence, there is at least one vertex r such that its adjacent edge $(g, r) \in A_i$ has higher value than the edge $(g', r) \in A^k_{i'}$. We exchange these goods resulting in the bundles $A'_i = A_i \backslash \{g\} \cup \{g'\}$, and $A'_{i'} = A_{i'} \backslash \{g'\} \cup \{g\}$.

In the corresponding new allocation in all these cases $A^\pi \backslash \{A_i, A_{i'}\} \cup \{A'_i, A'_{i'}\}$, we have $v(A'_{i'}) > v(A_{i'})$ and $v(A'_i) < v(A_i)$.

Until i' is not $\mathsf{EF1}$ with respect to i', we repeat this exchange or transfer. In the end we also have $v(A'_i) > v(A'_{i'})$, hence i is also $\mathsf{EF1}$ with respect to i'. Thus, both i and i' are $\mathsf{EF1}$ with respect to each other. QED.

Proof of Theorem 5.5.1. To prove this theorem, we observe that if an agent, i $\mathsf{EF1}$ envies agent j then all agents with value less than agent $i's$ value will $\mathsf{EF1}$ envy agent j. Thus, if there are any two agents, i and j such that i $\mathsf{EF1}$ envies j, then the agent with least value will $\mathsf{EF1}$ envy j. Now, we measure the progress of the algorithm using a tuple $(V, pairs)$ where V is the value of difference between the largest bundle that is $\mathsf{EF1}$ envied by some agent and the smallest bundle in the allocation. We use $pairs$ to denote the number of pairs of agents that have the difference V between them. The algorithm begins by computing an MSW allocation. While the allocation is not $\mathsf{EF1}$, we apply Lemma 5.5.1 to the largest bundle that is $\mathsf{EF1}$ envied by someone and the smallest bundle and make then $\mathsf{EF1}$ envy free with respect to each other. Therefore, we reduce the value of one largest bundle and increase the value of one smallest bundle. Thus, we cannot be creating new pairs with difference V between them and we have reduced at least one $\mathsf{EF1}$ pair of difference V. Therefore, in each iteration we either reduce V or we reduce $pairs$. Eventually, this shows that the algorithm progresses since there is a bound on V and number of pairs. and we

obtain an EF1+MSW allocation.

5.6 NON-EXISTENCE OF $(2/3+\epsilon)$-MMS ALLOCATION

In this section we negate the possibility of an algorithm that computes a better than $2/3$-MMS allocation for any fair division instance with OXS or higher valuation functions, by showing the following result.

Theorem 5.6.1. *There exists a fair division instance with* 2 *agents,* 4 *goods and* OXS *valuation functions of the agents for the goods that does not have a* $(2/3+\epsilon)$-MMS *allocation for any* $\epsilon > 0$.

Proof. We describe the bipartite graphs corresponding to the OXS valuation functions of the agents. Both agents have two vertices in the right parts of their graphs. Let the goods be denoted as $g_i, i \in [4]$, and the right vertices be denoted as r_1 and r_2. The weights of the non-zero weighted edges of the first agent's graph are as follows. $w(g_1, r_1) = w(g_2, r_2) = 2$, and $w(g_3, r_1) = w(g_4, r_2) = 1$. The second agent's graph's non-zero weighted edge weights are $w(g_1, r_1) = w(g_2, r_2) = 2$, and $w(g_4, r_1) = w(g_3, r_2) = 1$. The remaining edge weights in both graphs are zero.

It can be seen that the MMS value of both the agents is 3. MMS or higher valued bundles of the first agent are $\{g_1, g_4\}, \{g_2, g_3\}$, and $\{g_1, g_2\}$, while those of the second agent are $\{g_1, g_3\}, \{g_2, g_4\}$, and $\{g_1, g_2\}$. Any allocation that gives one agent an MMS or higher valued bundle removes at least one good from *all* such bundles of the other agent. Hence, a 1-MMS allocation does not exist. The highest possible value of a bundle smaller than MMS is 2, as all the goods have integer value. Hence, any allocation gives at least one agent a bundle of value at most 2, which is $2/3$-MMS, thus proving the theorem. QED.

5.7 HARDNESS OF APPROXIMATING MMS VALUE

In this section we will prove the following theorem.

Theorem 5.7.1. *The problem of computing the* MMS *value of an agent for an instance* $(n, \mathcal{M}, v)$ *is strongly* NP*-hard.*

Proof. We reduce the known strongly NP-complete problem 3-PARTITION to the problem of deciding if the MMS value of an agent for an instance $(n, \mathcal{M}, v)$ is at least T for a given $T \geq 0$, thus showing the later is also strongly NP-complete. The 3-PARTITION problem is as follows. Given a set of $3n$ integers with sum of all integers nT for some $T \geq 0$, one must decide if there

is a way to partition the integers in groups of 3 integers each such that the sum of each group is exactly T.

We create a fair division instance using an instance of 3-PARTITION as follows. Let there be n agents and $3n$ goods, one good corresponding to each integer in 3-PARTITION. The bipartite graph G_v corresponding to the identical OXS valuation function v of each agent is as follows. There are 3 vertices in the right part of the graph. Each good is connected to all three vertices and each edge weight is equal to the corresponding integer's value in the 3-PARTITION instance.

To see the correctness of the reduction, first note that as there are only 3 vertices in the right part of the graph, any matching M_v can have size at most 3. As there are exactly $3n$ goods, we can assume without loss of generality that any MMS allocation divides all the goods into n bundles of size exactly 3. Now each good's value in a bundle $\mathcal{S}$ of size 3 is its corresponding integer's value from the 3-PARTITION instance, hence the sum of all bundle values in an MMS allocation thus is nT. Thus the MMS value is at most T.

Now if the 3-PARTITION instance has a solution, then a partition of the goods in the corresponding fair division instance done according to this solution will give every agent a bundle of value T. If the 3-PARTITION instance does not have a solution, then every division of the integers into sets of size 3 has at least one set with sum less than T. As the set contains integers, this sum is at most $T-1$. Hence, the corresponding fair division instance also cannot generate a division of the goods into n bundles each of value greater than $T-1$, and the MMS value is at most $T-1$. QED.

CHAPTER 6: FAIR ROUTINGS

In this chapter, we study the problem of routing traffic with the dual optimization objective of optimizing both for travel time and envy-freeness. We will discuss in detail the following results.

- We begin with an impossibility result. We show that for certain networks, it is not possible to achieve optimality and no envy at the same time.
- We proceed with a hardness result, showing that it is NP-Hard to decide whether an optimal solution that is as fair as possible exists and that similarly it is NP-Hard to decide whether a near-optimal and fair solution exists.
- On the positive side, we first show that for a single origin-destination pair network, even with users with preferences, there exists an assignment of routes that is optimal in terms of total travel time and every driver has envy at most 3, meaning that there can't be a route of another driver that has cost less than $1/3$ times her own.
- Next we show that for the same setting, we can efficiently find a routing assignment that has total travel time at most twice the optimal travel time, and every driver has envy at most 4.
- Finally, we show that for the previously studied setting with users with identical preferences, we can find a routing assignment that is optimal and has envy strictly less than 3.

We also simulate experiments on real city graphs. In our experimental evaluation, we compare the performance of a fair algorithm with a natural greedy baseline that is similar to current solutions implemented by routing platforms. The outcomes of our experimental evaluation are as follows:

- We show that the solutions computed by our algorithm dominate the baseline, in the sense that for the appropriate parameters, our algorithm improves both the total cost and the worst case envy in the network.
- We extract plots that illustrate the trade-off between optimality and envy and how the volume of traffic impacts the results.
- We show that our algorithm is robust against the topology of the graph by studying cities with different topological characteristics.

6.1 PRELIMINARIES

Notation. We denote the set $\{1, 2 \cdots, n\}$ by $[n]$.

Congestion Networks Model. A *congestion network* models a set of users traversing a network of roads. A network of k *commodities* is formally specified by a directed graph G, k pairs of vertices (s_i, t_i), for $i \in [k]$, where each s_i is a source vertex and each t_i is a sink vertex in G, and a congestion function for each edge $c_e : \mathbb{Z} \to \mathbb{R}_{\geq 0}$, where if x_e users contain an edge e on their routes, then $c_e(x_e)$ is the cost to traverse e for each of these users. A *demand* for the network is specified by a k-length integer vector $(n_i)_{i \in [k]}$, where for every i, n_i is the number of users wanting to travel from s_i to t_i. Let $n = \sum_i n_i$, and A be the set of all users.

User preferences. We assume that drivers can have diverse preferences for road segments based on features such as time, distance, whether the segment is a highway, whether it has traffic lights, whether it precedes an unprotected left turn, etc. In the presence of such preferences, we might have envy between drivers even when their routes have identical delays. Consider for instance a driver who wishes to avoid highways who is routed on a highway. This driver would envy a route of the same travel time that does not include highways. We model user preferences as follows. Each edge on the network is associated with a set of binary *features*, intuitively to reflect if the corresponding road segment has some special characteristic or not. We consider a set of 7 features, like whether the edge is a highway segment, or if the edge has a toll booth on it, or if the road has a street light on it. Each edge e is associated with a 7-bit boolean vector y_e, where the k^{th} bit, denoted by $y_{(e,k)}$, is 1 only if e has the k^{th} feature. Each user i has a *preference* for each feature f, which is a value, denoted by $c_{(i,f)}$. Intuitively, users desire to travel routes with the smallest cost, and with segments containing features they like, and not containing features they dislike. We formalize this intuition by defining the preference cost of each user for every edge, denoted by $u_i : E \to \mathbb{R}$, where for any edge e,

$$u_i(e) = \sum_{k \in [7]} y_{(e,k)} * c_{(i,f)}.$$

A *routing assignment* is a vector $P = \{P_1, \cdots, P_n\}$, where P_i is a path in G from the source to the sink vertex of user i. For any routing assignment, the cost of every user i is defined as the congestion cost plus the preference cost of all edges on P_i, i.e.,

$$u_i(P_i) = \sum_{e \in P_i} c_e(x_e) + u_i(e),$$

where x_e is the number of users from A whose paths contain edge e.

We formalize the model with the following problem instance.

Definition 6.1.1 (Fair routing instance)**.** *A fair routing instance is denoted by a tuple* $\langle G, (s_i, t_i, n_i)_{i\in[k]}, C, (u_i)_{i\in[n]}\rangle$*, and represents a graph* G *with* k *source-sink pairs of vertices* (s_i, t_i)*, and* $n_i > 0$ *users associated with each pair;* C *is the set of linear congestion functions on the edges of* G*, denoted by* $c_e(x_e) = a_e x_e + b_e$ *for edge* e *with congestion* x_e*, with* $a_e, b_e \geq 0$*; each* u_i *is the preference function of user* i *for the edges of* G *as described previously. We will assume that every* $u_i(e)$ *satisfies* $|u_i(e)| \leq b_e$*, so that no user has an edge with negative travel cost to them.*

Envy-freeness. We define the following notion of fairness called *Envy-freeness* for any fair routing instance. For every routing assignment, the *envy ratio of user* i, denoted by $\mathsf{Er_i}$, is the ratio of the path assigned to i to the path with the least cost according to her cost function, i.e.,

$$\mathsf{Er_i(P)} = \max_{\mathsf{i'\in A}} \frac{\sum_{\mathsf{e\in P_i}} \mathsf{c_e(x_e)} + \mathsf{u_i(e)}}{\sum_{\mathsf{e\in P_{i'}}} \mathsf{c_e(x_e)} + \mathsf{u_{i'}(e)}}.$$

The *envy ratio of a routing assignment* is defined as $\mathsf{Er} = \max_{\mathsf{i}} \mathsf{Er_i}$. A routing assignment is called fair or envy-free (EF1) if its envy ratio is 1.

We define α approximately envy-free or α-EF1 routing assignments as those with Er at most α. Intuitively, users must prefer routing assignments with low envy ratio.

Optimality. A trivial observation is that for every (s_i, t_i) pair, if all of its associated n_i users are assigned the same $s_i - t_i$ path in G, then the outcome is a 1-EF1 routing assignment. But this assignment can have high cost to each user. Hence, we define optimal and fair assignments as those that upper bound the sum of costs to all the users. Formally, an *optimal routing assignment*, denoted by OPT, is one with the minimum sum of costs to all the users. A γ-approximately optimal or γ-OPT assignment, for any $\gamma \geq 1$, is one that has total cost at most γ times the cost of an optimal assignment.

With these notions, we define the following problem, which is the main focus of our paper.

Definition 6.1.2 (α-EF1 + γ-OPT.)**.** *Given a fair routing instance and constants* $\alpha, \gamma \geq 1$*, find an* α-EF1 *and* γ-OPT *routing assignment.*

6.2 ALGORITHMIC AND THEORETICAL RESULTS

In this section we design algorithms for the fair routing problem and prove various theoretical results. We first show that there are fair routing instances where *exactly* fair *and* optimal routing assignments do not exist. We then show that finding the fair and most near-optimal routing, or the most near-fair and optimal routing, is NP-hard. Finally, we prove constant approximation factors for both optimality and envy.

Theorem 6.2.1. *There exists a fair routing instance with a single source, single sink and* 2 *identical agents, and constants* $\epsilon, \epsilon' > 0$ *for which there are no* $\mathsf{OPT} + (1+\epsilon)$-$\mathsf{EF1}$ *or* $(1+\epsilon')$-$\mathsf{OPT} + \mathsf{EF1}$ *routing assignments.*

Proof. The instance is as follows. The network has one $s - t$ path with m edges for some odd m. Each edge on this path is called the *lower* edge between the corresponding pair of vertices, and the set of lower edges is denoted by L. Consecutive vertices on the path are additionally connected by a second edge. We call all these second edges the *upper edges*. There are 2 users with identical cost functions as follows. Each upper edge has a linear cost function al and the i^{th} lower edge for each i has the cost function $a_i l$, where l is the congestion on the edges, and a and a_i are constants specified shortly.

We will prove the following two claims about this instance. First, every optimal routing assignment assigns disjoint routes to the users, and every envy-free assignment has at least one common edge assigned to both users. For this, we impose that the a_is are distinct, the coefficients in the cost functions satisfy

$$\frac{\max_i a_i}{\min_i a_i} < 6 \text{ and } \frac{a_i}{3} < a < 3a_i \text{ for every } a_i.$$

We now prove the first claim. As a user can reach any vertex (except s where they start) only by first reaching its previous vertex, it will suffice to prove that for any given consecutive pair of vertices, any optimal assignment sends one user on the lower edge and one on the upper edge. The cost of sending both users on a single edge is $4a$ or $4a_i$, while that of sending users on distinct edges is $a + a_i$. As $a > a_i/3$, $4a_i > a_i$, and as $a < 3a_i$, $4a > a + a_i$, hence the optimal strategy is to send both users on distinct edges.

Next, we prove that an envy-free assignment cannot route users on disjoint paths. Suppose for contradiction this were not the case. Then let P_1 and k be the set of lower edges and the number of upper edges assigned to the first user. Then the costs of both users are $\sum_{e \in P_1} a_i + ka$ for the first user and $\sum_{e \in L \setminus P_1} a_i + (m-k)a$ for the second. As this assignment is envy-free, we have

$$\sum_{e \in P_1} a_i + ka = \sum_{e \in L \setminus P_1} a_i + (m-k)a.$$

One can always choose a rational number a to ensure that none of the first m multiples of a is equal to any possible difference in costs from the lower edges. This implies that the costs from the upper edges (and from the lower edges) should be equal for both the users. But as m is odd, the costs from the lower edges cannot be equal, giving a contradiction.

Thus, every envy-free assignment has at least one consecutive pair of vertices where both users travel through the same edge between this pair. The cost to the users in such an assignment from

each common edge is at least $\min\{4a, 4a_i\}$. As the optimal assignment gives a cost of $a + a_i$ from each such pair, the difference in costs between any optimal and envy-free assignments is,

$$\sum_i \min\{4a, 4a_i\} - (a + a_i) > 0.$$

Alternatively, each envy-free assignment has a greater cost than any optimal assignment, and each optimal assignment has greater than 1 envy. QED.

In light of Theorem 6.2.1, we relax our aim and investigate the problem of finding α-EF1 + OPT or EF1 + γ-OPT routing assignments for the best possible α and γ for each fair routing instance. Formally, we define the following problems.

Definition 6.2.1 (Near-fair and optimal routing)**.** *Given a fair routing instance, the aim is to find an* α-EF1 + OPT *routing assignment for the lowest* $\alpha \geq 1$ *for which an* α-EF1 + OPT *assignment exists for the instance.*

Definition 6.2.2 (Fair and Near-optimal routing)**.** *Given a fair routing instance, the aim is to find an* EF1 + γ-OPT *routing assignment for the lowest* $\gamma \geq 1$ *for which an* EF1 + γ-OPT *assignment exists for the instance.*

The decision versions of the above problems are to decide if a given α (or γ) is the optimal value for which an α-EF1 + OPT (respectively EF1 + γ-OPT) routing assignment exists.

Theorem 6.2.2. *The decision versions of the Near-Fair and Optimal Routing and the Fair and Near-Optimal Routing problems are NP-hard.*

Proof. We reduce from the NP-complete Equal-Cardinality Partition problem, which is defined as follows. Given a set of rational numbers $S = \{e_1, \cdots, e_n\}$, decide if there is a subset of half of these whose sum is half of the sum of all the numbers. We assume without loss of generality that $\max_i e_i / \min_i e_i < 6$, as one can add a sufficiently large number to each element e_i to achieve this without changing the solution set.

For an instance of this problem, the reduced fair routing instance for both the problems is as follows. Consider a network with 2 users with identical cost functions, a single source s and a single sink t, and one $s - t$ path of n edges, referred as the *lower* edges. Each consecutive pair of vertices also has another edge connecting them; call this set of alternate edges the *upper* edges. The cost function of each lower edge is $e_i l$, and of each upper edge is el, where l is the congestion on the edges, and e is a constant that satisfies $e_i/3 < e < 3e_i$ for every e_i.

From the same reasoning as in the proof of Theorem 6.2.1, we say that every optimal routing assignment assigns disjoint paths to both users. From a similar reasoning from the same proof, we

can say that an envy-free assignment can assign disjoint paths to both users only if the cost to the first user from the set of lower edges is equal to the cost to the second from the same. Thus, an envy-free and optimal assignment exists if and only if there is a solution to the Equal-Cardinality Partition problem. Thus, the α (or γ) for the Near-fair and Optimal (respectively fair and near-optimal) routing problem is 1 if and only if Equal-Cardinality Partition has a solution. QED.

We complement the hardness results by showing several positive results. First we show the existence of a 3-EF1 + OPT routing assignment for any single source single sink network in our setting. [78] show a similar result for the special case of users with identical cost functions. Formally, we prove the following theorem.

For this we use the concept of envy cycle elimination from fair division [17], which briefly is the following. Suppose we construct a directed graph where the nodes represent users, and there is a directed graph from any user i to i' if the cost of the route assigned to i' is smaller than that of i according to the cost function of user i. Then in time polynomial in the number of users, we can shuffle the routes so that the envy graph of the resulting assignment has no cycles, and the total cost has not increased. We will call this procedure *envy-cycle elimination* and use it repeatedly in our main algorithm. Note that this procedure does not alter the routes assigned, and only shuffles them among the users.

After applying the envy-cycle elimination method, we know that the envy between any pair of agents in the resulting routing assignment is one-sided, or formally,

Claim 6.2.1. *If for a pair of users i, j, the envy ratio of their routes is higher than 3 according to i, then the envy ratio according to j is less than 1.*

We will assume without loss of generality that Claim 6.2.1 is true for any routing assignment of a setting of users with preferences, as otherwise envy can be resolved by applying the envy cycle elimination procedure.

Theorem 6.2.3. *Any optimal routing assignment of a single source single sink network with users with preferences has envy less than 3.*

Proof. Suppose for contradiction there is a user i with Er at least 3 for some user j's route. Denote the routes of i and j by P_i and P_j, the set of users assigned any edge e on these paths by A_e with $|A_e| = x_e$, and let the total cost of each route P_k for user k by t_k. Then consider the routing assignment obtained by switching the path of i to P_j. Let us compute the difference in costs of the two assignments. As route assignments change only on edges along P_i and P_j, after re-assigning the highest cost user to the lowest cost route, the change in the total sum of costs of all users is,

$$
\begin{aligned}
&\sum_{e\in P_i}\sum_{k\in A_e}(a_e x_e + b_e + u_k(e)) - \sum_{k\in A_e\setminus\{i\}}(a_e(x_e-1)+b_e+u_k(e)) \\
&+\sum_{e\in P_j}\sum_{k\in A_e}(a_e x_e + b_e + u_k(e)) - \sum_{k\in A_e\cup\{i\}}(a_e(x_e+1)+b_e+u_k(e)) \\
&=\sum_{e\in P_i}(a_e x_e+b_e)x_e - \sum_{e\in P_i}(a_e(x_e-1)+b_e)(x_e-1) + u_i(e) \\
&+\sum_{e\in P_j}(a_e x_e+b_e)x_e) - (a_e(x_e+1)+b_e)(x_e+1) - \sum_{e\in P_j}u_i(e) \\
&=\sum_{e\in P_i}(a_e(2x_e-1)+b_e) - \sum_{e\in P_j}a_e(2x_e+1)+b_e \\
&+\sum_{e\in P_i}u_i(e) - \sum_{e\in P_j}u_i(e) \\
&\geq \sum_{e\in P_i}(a_e x_e+b_e) - \sum_{e\in P_j}3a_e x_e + b_e + 0 - \sum_{e\in P_j}b_e \\
&\geq \sum_{e\in P_i}(a_e x_e+b_e) - \sum_{e\in P_j}3(a_e x_e+b_e) \\
&\geq t_i - 3t_j > 0. \qquad (6.1)
\end{aligned}
$$

The third to last inequality follows from the relations $|u_i(e)| \leq b_e$ for every user i and edge e. Note that the equation is true even if t_j is assumed to be the cost of user i for j's route. We will use both interpretations of t_j in the subsequent discussion. Equation (6.1) shows that the new routing assignment has lower cost than the optimal assignment, a contradiction, thus completing the proof. QED.

Next, we consider the setting when the preference costs of the users are non-negative, i.e., $u_i(e) \geq 0$ for every user i and edge e, and show the following.

Theorem 6.2.4. *There is an efficient algorithm for computing a* 4-EF1$+2$-OPT *routing assignment exists for a single source single sink network and users with non-negative preference costs.*

The starting point of the algorithm that computes a 4-EF1 $+$ 2-OPT routing assignment is the algorithm of [78], that computes an optimal routing assignment for the identical users setting. We prove that with the correct cost functions, this algorithm returns a 2-OPT assignment for our setting.

Lemma 6.2.1. *There is a polynomial time algorithm that computes a* 2-OPT *routing assignment for a single source single sink network with non-negative preferences.*

Proof. The algorithm first reduces the instance to an identical users setting by changing all preference costs, i.e., $u_i(e)$ for every user i and edge e, to zero. Let us denote the original and reduced

Algorithm 12: (< 3)-EF1 + OPT routing assignment for single source-sink network and users with identical linear costs

Input : $\langle G, (s, t, n), C, (u_i)_{i \in [n]} \rangle$
Output: (< 3)-EF1 + OPT routing assignment

Initialize $r = 0$ // iteration number
$P^0 =$ OPT routing assignment // Algorithm from [78]
while Er ≥ 3 **do**
  $r = r + 1$
  $i =$ any user with highest cost in P^{r-1}
  $j =$ any user with lowest cost in P^{r-1}
  $P_i^r = P_j^{r-1}$, $P_k^r = P_k^{r-1}$, $\forall k \neq i$ // re-assign path of highest cost user to lowest cost path
Return P^r

instances by I^p and I^{id} respectively. It then applies the algorithm of [78] to compute an optimal routing assignment for I^{id}, and assigns the paths obtained arbitrarily to any user in I^p. Let us analyze the total cost of this assignment.

The total cost according to the cost functions in I^{id} is $C = \sum_e (a_e x_e + b_e) x_e$. As all preference costs in I^p are non-negative, and C is the optimal cost for I^{id}, the optimal cost for I^p is at least C. The total cost of the routing assignment obtained for I^p is $\sum_e \sum_{i \in A_e} (a_e x_e + b_e + u_i(e)$, where A_e is the number of users assigned edge e. This cost is at most:

$$\sum_e \sum_{i \in A_e} (a_e x_e + 2b_e) \leq \sum_e (a_e x_e + 2b_e) x_e \leq 2 \cdot \sum_e (a_e x_e + b_e) x_e = 2C.$$

Thus this assignment is 2-OPT for I^p. QED.

We can now look at the 2-OPT routing assignment obtained as described in the proof of Lemma 6.2.1, and assume without loss of generality that Claim 6.2.1 is true for the same. Suppose this assignment has envy higher than 4. The next lemma describes a process to reduce the envy between users and decreasing the total cost in the process.

Lemma 6.2.2. *Any k-EF1 routing assignment can be converted into a 4-EF1 routing assignment without increasing the total cost of all users in time $O(k^2 poly(n))$.*

Proof. The algorithm to convert a k-EF1 assignment to a 4-EF1 routing assignment is to repeatedly consider any pair of users i, j with the highest envy ratio among all pairs, according to one of them from Claim 6.2.1, say i. Then switch i's route to the route of j. From equation (6.1), while the envy of the assignment is higher than $k - 1$, the total cost of all users reduces by at least:

$$t_i - 3t_j \geq (k - 4)t_j.$$

Now if t_j* denotes the cost of the user with the smaller cost route after any such iteration, and t_u the cost of any user u in the initial routing assignment, we have that $t_j* \geq t_u$ for some u. This is because either each iteration introduces a new user as the smaller cost user, or some user from a previous round repeats. Hence, while the envy of the routing assignment is higher than $k-1$, the total cost of all users decreases by at least:

$$(k-4)t_u \geq (k-4)\min_u t_u,$$

in each iteration. As the envy of the initial routing assignment is at most k, its total cost is at most $kn \min_u t_u$. Hence, in at most $O(nk)$ iterations, the cost will be less than

$$(n-1)\min_u t_u + (k-1)\min_u t_u,$$

and the envy reduces to below $k-1$. Hence, in:

$$n \cdot (\sum_{k \geq 4} k) = O(nk^2)$$

iterations, the envy of the assignment reduces to smaller than 4. After each iteration, we run the envy cycle elimination procedure to ensure Claim 6.2.1 remains true. QED.

Lemmas 6.2.1 and 6.2.2 together establish Theorem 6.2.3.

Our final result considers the setting of identical users. We prove the following theorem.

Theorem 6.2.5. *Algorithm 12 finds an* α-EF1+OPT *routing assignment for* $\alpha < 3$, *a single source single sink network and users with identical cost functions.*

We will prove this theorem in the rest of this section. The Algorithm starts by computing an optimal routing assignment. [78] show that such an assignment is 3-EF. We first show in Lemma 6.2.3 that the routing assignment obtained after every iteration of the while loop is also 3-EF1. This means that either the assignment has Er equal to 3 and the algorithm continues, or has Er less than 3 and is returned.

Lemma 6.2.3. *The routing assignment obtained after every iteration of the while loop in Algorithm 12 is* 3-EF1 + OPT.

The proof follows the reasoning of the proof of Lemma 6.2.3 to show that if the routing assignment at the beginning of an iteration is optimal, then so is the one at the beginning of the next. As each routing assignment is optimal, from [78], it is also 3-EF1.

Thus, the routing returned by Algorithm 12, if it converges, has Er less than 3. It now remains to show this process converges in polynomial time. For this, we prove the following lemma.

Lemma 6.2.4. *For every iteration r of Algorithm 12, the cost of all users in the routing assignment P^r are in $[t, 3t]$.*

Proof. Note that in P^0, every user has cost in $[t, 3t]$. In any iteration r, after switching the path of user i to the path of user j, any user whose cost reduces, say i', must share at least one edge with i in P^{r-1}, and any user whose cost increases, say j', shares at least one edge with j in P^{r-1}. The costs to i' and j' from the shared edges must be non-zero, as the change in their costs is only due to the change in congestion on the shared edges. We will show that the cost of such an i' in P^r is higher than t, and that of j' in P^r is lower than $3t$.

First, when any two users share some edges on their routes in any assignment, then following the analysis of equation (6.1), we can show that the ratio of their costs from the unshared edges is at most 3, else we can switch the route of the higher cost user to the other and reduce the total cost.

Now consider i'. Let t_s be their cost from the edges shared with i in P^{r-1}, and let $t_{i'}$ and t_i respectively be their costs from the unshared edges. We want to find the minimum cost of i' in P^r. We know from the above reasoning that $t_i/t_{i'} \leq 3$, and as i switched paths, $t_i + t_s = 3t$. The congestion on the edges of i' shared with i reduces by 1. This can reduce the cost of i' from the shared edges by at most half, hence the new cost of i' is at least $t_{i'} + t_s/2$. Now $t_{i'} + t_s/2 \geq t_i/3 + t_s/2$, as $t_i/t_{i'} \leq 3$, and $t_i/3 + t_s/2 > (t_i + t_s)/3 = t$, as $t_i + t_s = 3t$. Thus, i' has cost higher than t in P^r.

By a similar analysis we can show that the cost of j' is lower than $3t$ in P^r, thus completing the proof. QED.

The proof of the above lemma also implies the following.

Corollary 6.2.1. *The number of users with cost $3t$ and those with cost t reduce by 1 in every iteration of Algorithm 12.*

Lemmas 6.2.3 and 6.2.4 imply that if the algorithm does not converge after some iteration, the corresponding routing assignment has at least one user with cost t and one with cost $3t$. Corollary 6.2.1 implies that the number of users with these costs reduce by 1 in every iteration. As there are n users, Algorithm 12 converges in at most $n/2$ iterations, completing the proof of Theorem 6.2.4.

6.3 EXPERIMENTAL EVALUATION

In this section, we use real road networks to construct instances where we evaluate the performance of a fairness-aware algorithm against a simple greedy baseline. First we describe the specifics of the algorithm, then we describe the data used to construct our experimental instances, and finally we close the section with the results of our evaluation.

6.3.1 Algorithms

Fair Algorithm. Our fairness-aware algorithm relies on solving a quadratic integer program (QIP) that optimizes the total cost to all users subject to given envy constraints. As such a program can be too expensive to run, we first precompute a small set of efficient s-t paths using well-studied penalty-based alternate path generation algorithm [95, 117], then solve another QIP that has the same objective but is restricted to assigning each user one of the pre-computed paths. The number of routes we produce per s-t pair is typically between 5 and 10, depending on the distance between the origin and destination. We note that the penalty method is considered to be the gold standard for producing alternative routes of high quality in road networks. The algorithm proceeds by iteratively producing a shortest path from origin to destination and applying a multiplicative penalty to each selected segment. The edge costs we use during alternative route computation are the free-flow costs, $c_e(0) = b_e$ for each e, and the multiplicative penalty we apply each time an edge is used is 1.2.

Our QIP is as follows. We denote the set of users by U, the set of paths by P and the union of edges on these paths by E^p. Also let $v_u(e)$ denote the preference cost of user u for edge e, and α denote the maximum tolerable envy of the assignment. Our primary variables are $x_{(u,p)}$, which are indicator variables for whether a path p has been assigned to user u. To ease readability, we add variables (with equality constraints) $c_{(u,u')}, \forall u, u' \in U$ for the cost of user u for the path assigned to user u', y_p for $p \in P$ which compute the number of users assigned to a path p, and q_e for $e \in E^P$ denoting the number of paths *assigned in the final solution* that traverse through edge e. In the simulations we have removed these variables.

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{u \in U} c_{(u,u)} \\
\text{Subject to } & c_{(u,u')} = \sum_{p \in P} x_{(u',p)} \sum_{e \in p} (a_e q_e + b_e + v_u(e)), \quad \forall u, u' \in U \\
& c_{(u,u)} \leq \alpha \cdot c_{(u,u')}, \quad \forall u, u' \in U \\
& \sum_{p \in P} x_{(u,p)} = 1, \quad \forall u \in U, p \in P \\
& \sum_{u \in U} x_{(u,p)} = y_p, \quad \forall u \in U, p \in P \\
& \sum_{p \in P} \mathbb{1}_{(e \in p)} \cdot y_p = q_e, \\
& x_{(u,p)} \in \{0, 1\}, \quad \forall u \in U, p \in P
\end{aligned}
$$

Naturally, the QIP is not guaranteed to find an optimal solution in polynomial time, however as we will see a good approximate solution is extracted, which proves sufficient for practical purposes.

Greedy Baseline. Our baseline is a greedy algorithm that simulates current navigation systems. We route requests one by one, and assign each driver to the path that is optimal with respect to her preferences under current conditions, by means of a simple Dijkstra search. The edge costs are updated after each request we route.

6.3.2 Data

Graphs We extract graphs from Open Street Maps [118] for New York, Tokyo, Delhi, and the San Francisco Bay Area (including the whole region from Berkeley to San Jose). Each region we chose exhibits a different property. For example, Tokyo consists of very dense and highly connected network; whereas the Bay Area consists of dense areas (city centers) connected rather sparsely. We believe our choices demonstrate the versatility of our methods over different network structures.

For each road and intersection, we extracted whether 1. it has a traffic light, 2. it has a stop sign, 3. it is lit, 4. there is a toll, 5. it has a sidewalk, 6. it is on a highway; along with its distance, number of lanes and maximum speed allowed.

Given the road data, we construct our graphs in the following way:

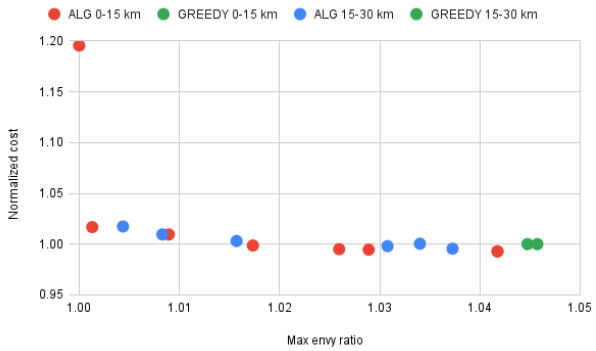
- We add one node per road in each available direction;
- We add a directed edge from the node representing one road segment to the other if the corresponding turn is permitted.
- The cost of each directed edge is calculated as the sum of costs for the starting road segment and the corresponding turn.

The sizes of each graph we obtained are summarized in the following table.

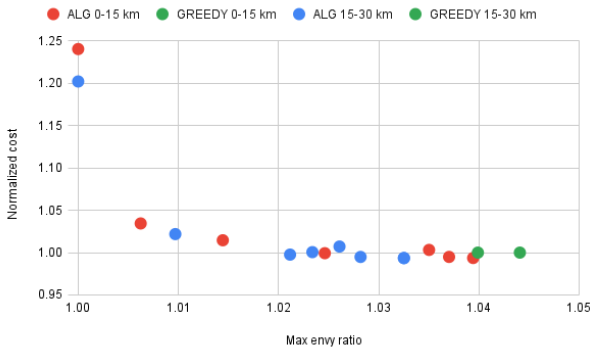
Region Name	# of Nodes	# of Arcs
Bay Area	4.7M	10.1M
Delhi	1.3M	3.3M
New York	2.5M	5.5M
Tokyo	11.3M	26.8M

The costs per roads are calculated using the functional form of the Bureau of Public Roads [119], similar to [85, 120]. Specifically we set:

$$c_e(x) = \frac{t_e^f}{\kappa_e} x + t_e^f,$$



(a) Delhi



(b) New York City

Figure 6.1: Plots of our algorithm showing a trade off between fairness and optimality. The x axis shows a decreasing fairness requirement. Each point, say 1.02, indicates that we only allowed route assignments where the maximum envy ratio was at most 1.02. The y axis shows the cost of the solution (normalized by the cost of baseline, also shown in the figure). While it is natural to expect that the cost will reduce with a more relaxed constraint, our findings show that in all cases, for small envy ratios, the cost is near optimal. In other words, almost fair and near optimal solutions exist and can be computed.

where t_e^f is the time to cross the edge when the road is empty, i.e., the free-flow travel time, and κ_e is the capacity of the street, defined as the number of lanes multiplied by the free-flow speed.

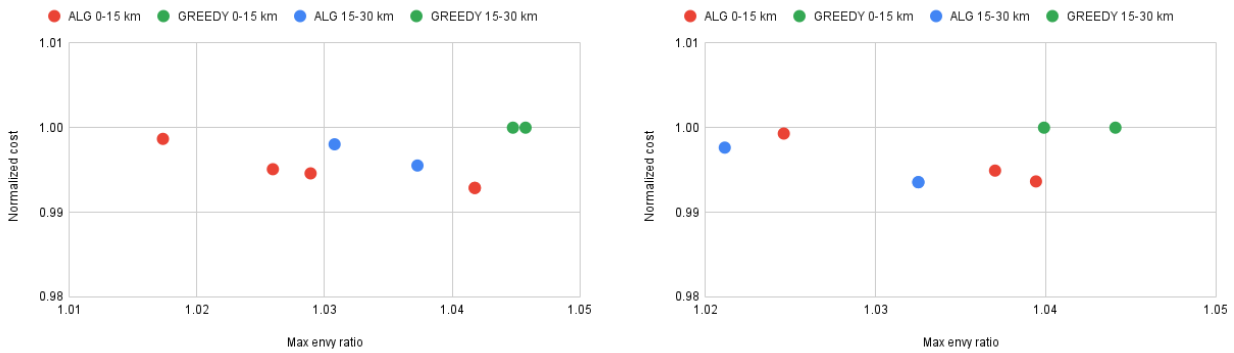
Traffic Flows To generate traffic flows, we create uniformly random routing requests between pairs of locations in the cities. We set the preference of each user randomly: Some may want to avoid tolls and prefer lit roads, whereas some may want to avoid left turns but prefer highways. The typical number of requests between a pair of locations is approximately $1,000$ vehicles.

Parameter Values. For each city, we run our fair algorithm with 8 envy-freeness target values: 1 to 1.1 in steps of 0.02,, then 1.25 and 1.5, and record the corresponding cost. Note that the greedy baseline is not parameterized by an envy-freeness target value, and instead naturally induces some maximum envy, which we also record, along with the solution's cost. We run the experiment in each city, for two classes of trips: 0 to 15 km trips and 15 to 30 km trips. In each class we generate 5 instances.

6.3.3 Results

The main outcome of our evaluation is that over a number of cities with diverse road network architectures, our fair algorithm outperforms the natural (and typically in use) greedy baseline in both dimensions: fairness and cost, when the envy-freeness parameter is chosen correctly.

We present plots that show the baseline as a point in the envy vs cost two-dimensional plain



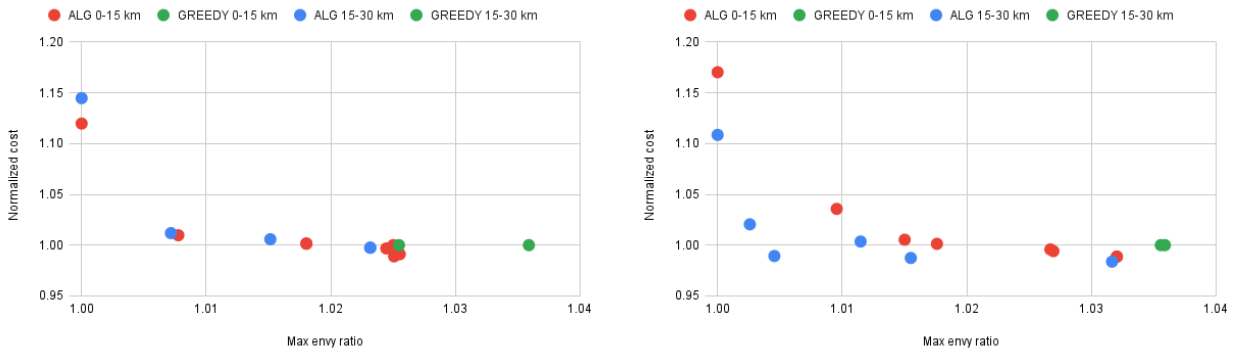
(a) Delhi

(b) New York City

Figure 6.2: Plots comparing the performance of our algorithm with the baseline over two dimensions: fairness and optimality. The x axis shows several envy ratio constraints, and the y axis shows the the average cost of the solution (normalized by the cost of the baseline). The baseline is a single point as it is not parameterized by an envy target, in contrast to our algorithm. Our findings show that in all cases, our algorithm performs better than the baseline in terms of both the fairness and optimality. The points that show the best solutions of our algorithm, that is, those that are most envy-free and have cost lower than the baseline, have (envy, cost) values $(1.017343, 0.9986738758)$ for Delhi and $(1.021151, 0.997642824)$ for New York city.

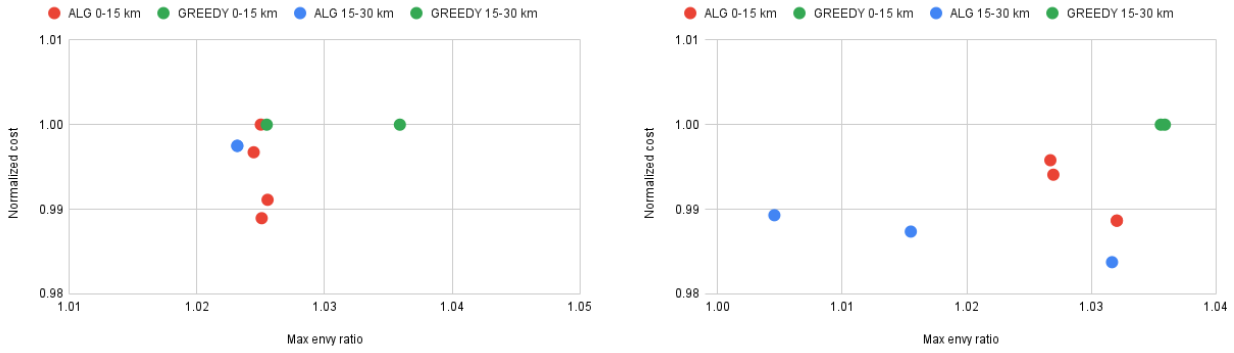
and our algorithm as various points, depending on the envy target. Our points are given by trips aggregated in buckets based on their length in km. The results for the cities Delhi and New York City, are shown in Figures 6.1 and 6.2. The full results are in Figures 6.3 and 6.4.

In these plots, points that are lower and to the left are better than points that are higher and to the right. In each one of the cities, we observe gains by applying our fairness-aware algorithm as opposed to the greedy baseline when the algorithm's envy parameter is chosen to be higher than 1.02. We also observe the trade-off between optimality and fairness by the curve of the algorithm. The cost of routing assignments with envy parameters stricter (that is, smaller) than 1.02 increases rapidly, indicating that an ideal cost-fairness trade-off would be to sacrifice about 2% fairness to keep the cost of each user reasonably low. In general, slightly relaxing fairness can give strong gains in terms of cost, which become diminishing rather quickly and further relaxations on the max envy ration have a relatively small impact on the total cost.



(a) San Francisco **(b)** Tokyo

Figure 6.3: Plots of our algorithm showing a trade off between fairness and optimality. The x axis shows a decreasing fairness requirement. Each point, say 1.02, indicates that we only allowed route assignments where the maximum envy ratio was at most 1.02. The y axis shows the cost of the solution (normalized to the cost of baseline, also shown in the figure) for each requirement. While it is natural to expect that the cost will reduce with a more relaxed constraint, our findings show that in all cases, for small envy ratios, the cost is near optimal. In other words, almost fair and near optimal solutions exist and can be computed. We show the corresponding plots of the greedy algorithm too for comparison, but illustrate that in more detail in Figure 6.4.



(a) San Francisco **(b)** Tokyo

Figure 6.4: Plots comparing the performance of our algorithm with the baseline over two dimensions: fairness and optimality. The x axis shows several envy ratio constraints, and the y axis shows the the average cost of the solution (normalized by the cost of the baseline). Each point shows the cost of the optimal solution among those that satisfy the corresponding envy constraint. The baseline is a single point as it is not parameterized by an envy target, in contrast to our algorithm. Our findings show that in all cases, our algorithm performs better than the baseline in terms of both the fairness and the total cost of the solution. The optimal (envy,cost) values obtained by our algorithm are $(1.0231634, 0.9974964011)$ for San Francisco and $(1.004555, 0.9893005474)$ for Tokyo.

CHAPTER 7: FAIR ROUTES WITH ALTERNATE SUGGESTIONS

In this chapter, we explore the routing traffic setting, where drivers arrive in a streaming fashion, and their cost functions for the roads may dynamically change. Our goal is to assign paths to each user that guarantee fairness and economic efficiency.

This setting models web-based routing platforms, that address the important every-day problem of navigating drivers in the road network and have reached numbers of users that are in the billions. The expectations from the users perspective is that the platform will provide them with a latency minimizing route to the location they want to reach, and is fair to them. First, even without the fairness requirement, to find such a latency minimizing route, the platform needs to maintain accurate estimates of the availability of road segments as well as their expected delays. This is information that, by the nature of road networks, is dynamically changing. These changes, which can be attributed to events such as road closures and accidents or even to natural perturbations of traffic demands, significantly complicate the task of recovering the optimal route between endpoints.

The complexity of finding a shortest path in the presence of dynamic costs stems from various reasons. For one, a dynamic shortest path computation is much more demanding in terms of latency than a shortest path on static costs. Standard pre-computation methods employed to make road networks amenable to fast shortest path search [121] are not applicable in the dynamic setting, whereas those that are [122, 123] need some time to update the costs and always have some lag in reflecting the true conditions on the road network. A second complexity stems from the fact that accurately predicting the delay of a route is by definition an easier task than accurately predicting the delay of each one of its individual road segments (since accurate segment delay predictions yield accurate route delay predictions). Path search requires the latter whereas evaluating a path requires the former, meaning that platforms might be able to produce good estimates for the routes but can have a hard time actually finding the best ones.

Our main thesis is that, utilizing past data on the demand between origin-destination pairs in the network and on the historical travel times observed on road segments, a routing platform can get past the challenges outlined in the previous paragraph and can provide high quality routing to its users. Such data is amply available to routing platforms. We show both with analytical results and with experiments on real city graphs that the solution reached by our algorithm is very close to the one that can be achieved with perfect travel time predictions. Our approach relies on pre-constructing a set of alternative routes for each user that we anticipate to see in the network. We do not require knowledge of specific characteristics of the user, just an estimate on the number of them that will request to be routed from a given origin to a given destination. Our algorithm will then assign a route from these pre-computed alternatives to the user upon arrival in the system.

7.1 PRELIMINARIES

We will denote by $[k]$ the set $\{1, 2, \ldots, k-1, k\}$.

Model. We consider the setting where a set of n users wish to travel between a given pair of nodes in a road network in a congestion aware manner. It is known that only a constant fraction $c \in (0, 1]$ of the users will be *active*. Our aim is to assign a small set of paths to each user such that for any cn-sized subset of users, if these users appear in any order and greedily choose the shortest unassigned path from their pre-assigned paths, the resulting path assignment is comparable to an optimal assignment of paths that could be computed after the set of users is known.

We formalize this setting using the following definitions.

Definition 7.1.1. *[Routing instance] This is a tuple $\langle n, s, t, C\rangle$, where s, t are nodes in G, and n is the number of users who wish to travel from s to t. C is the set of non-negative monotone cost functions $c_e : n \to \mathbb{R}_{\geq 0}$ for each edge e in the road network G.*

Definition 7.1.2. *[Routing Algorithm] Given an instance $\langle n, s, t, C\rangle$, and a subset of cn users for some constant $c > 0$, an algorithm that generates a set of cn paths from s to t, one for each user, is called a routing algorithm.*

Definition 7.1.3. *[Alternate route assignment] Given an instance $\langle n, s, t, C\rangle$, an alternate route assignment is a set of n sets of paths from s to t, denoted by $R = \{R_1, R_2 \ldots, R_n\}$, where the i^{th} set R_i is assigned to the i^{th} user.*

Definition 7.1.4. *[k-ALTALGO] An algorithm that generates a routing by first creating an alternate route assignment to n users such that every user is assigned at most $O(k)$ paths, then a routing using alternates as per Definition 7.1.5, is called* k-ALTALGO.

Definition 7.1.5. *[ROUTFrmALT] Given an alternate route assignment R to n users, and a subset of cn users for some constant $c > 0$, an algorithm that generates a routing by first ordering the users arbitrarily, then giving to every i^{th} user the shortest path from R_i that has not been assigned to any of the first $i-1$ users, or if all paths in R_i are already assigned then the shortest path in R_i, is called* ROUTFrmALT. *The output of the algorithm is denoted by $r = \{r_1, r_2, \ldots, r_N\}$, where $N = cn$.*

In this paper, we design k-ALTALGO algorithms for some small k. We compare the routing obtained from the ROUTFrmALT that use these k-ALTALGO with an optimal routing algorithm using the following notions of optimal algorithms and approximation parameters.

Optimal routing. We consider an optimal routing as that obtained by a greedy algorithm that iteratively computes the shortest path for all the n users. Denote the optimal set of paths for all n users as $OPT = \{P_1 \ldots, P_n\}$. These paths need not be distinct. Note that we can assume the

preference relation $P_i \succ P_i$ for any two paths where $i < j$, by ranking the n paths after they are computed according to the users' identical cost functions. Then for every $c \in (0, 1]$, the first cn paths of P are the optimal assignment for any cn-sized subset of users.

Approximation parameters.

Rank. If the ROUTFrmALT that uses a k-ALTALGO results in all the users choosing paths ranked at most $(\alpha \cdot cn)$ for *any* cn-sized subset of the users in expectation, then we say that this k-ALTALGO is an α-rank approximation of the optimal algorithm.

Note that higher the value of k, easier it is to ensure a good rank approximation guarantee. For example, if $k = n$, then we can assign all paths to every user, and any subset of cn users will be able to choose one of the top cn paths. We show such an assignment exists with high probability for a particular ordering of the users for $k = \log n$.

Fairness. We define two notions of fairness. First is using what we call the *envy ratio,* which is defined as the ratio between the worst and the best routes of any route assignment. If the envy ratio of the assignment produced by ROUTFrmALT is β, then we say its corresponding k-ALTALGO is a β-envyfree algorithm.

Our second notion measures the fairness of the assignment of alternate routes as follows. Consider the alternate routes assigned to any user i. We say this user envies some other user j up to at most p paths, denoted by EFp (read as *envy-free up to p paths*) if after removing their commonly assigned paths, the best path of i is no worse than the p^{th} best path of j. We call a k-ALTALGO EFp if it generates an assignment that is EFp for each user.

In this paper, we show two algorithms, a $\log$ n-ALTALGO deterministic algorithm which gives a constant factor like rank guarantee in expectation. Then with a slightly higher number of paths and using randomization, we obtain O$(\log^2 \mathsf{n}/\epsilon)$-ALTALGO, which gives a PTAS like $(1+\epsilon)$-rank guarantee with high probability. Both algorithms also ensure envy-free fairness guarantees. We discuss these in the following sections.

7.2 ANALYZING USERS DETERMINISTICALLY

In this section we will prove the following theorem.

Theorem 7.2.1. *For any routing instance $\langle n, s, t, C\rangle$, Algorithm 13 is a $4(1 + 2/e)$-rank approximation to OPT.*

Algorithm 13 works as follows. We first compute $2^t - 1$ optimal paths, for $2^t - 1$ the smallest integer higher or equal to n. Then we have two stages of assigning alternate routes to each user. The first is the deterministic stage. Here we first create 2^t sets of alternates as follows. Path P_1 is assigned to all the n users. Then for every i, the next 2^i paths are each assigned to a distinct set of

Algorithm 13: ($\log n$)-ALTALGO

Input : $\langle n, s, t, C \rangle$
Output: Alternate routing assignment that assigns at most $O(\log n)$ paths to each user

$2^t :-$ smallest power of 2 higher than n
Compute $2^t - 1$ **OPT paths**
$OPT :- \emptyset$
$T :- \{0\}^m$ `// initialize congestion of every edge` e `to` $T_e = 0$
for $i \in [2^t - 1]$ **do**
    Find shortest path P_i in G where each edge has weight $c_e(T_e)$
    Add P_i to OPT
    for *edge* e *in* P_i **do**
        $T_e \leftarrow T_e + 1$
Assigning alternate routes:
1. Deterministically assign $O(\log n)$ **paths**
for $i \in [2^t - 1]$ **do**
    $R_i \leftarrow \emptyset$ `// initially no paths assigned`
    for $j \in [t]$ **do**
        $l \leftarrow 2^{j-1} + \lfloor \frac{i \cdot 2^{j-1}}{n} \rfloor$
        $R_i \leftarrow R_i \cup \{P_l\}$
for $i \in [2^t - 1] \backslash [n]$ **do**
    $R_{i-n} = R_{i-n} \cup R_i$ `// assign any excess alternate sets to users`
Return $R = \{R_1, R_2 \cdots, R_n\}$

2^{t-i} users. After these sets are formed, we assign the j^{th} and $((j+n) mod (2^t - 1))^{th}$ sets to the j^{th} user. The next phase then randomly picks k users for every user u, and assigns the union of their deterministically assigned sets to u.

If n is not of the form $2^t - 1$, the algorithm creates extra paths and assigns more paths to each user, which will help in our analysis. Hence, for worst case analysis, we will assume $n = 2^t - 1$ for some t. We now set up some notation for analyzing this algorithm.

Notation

Bipartite Graph Matching. Let us call a user *active* if they are a part of the selected cn sized subset of users, and call a path *active* if it is assigned to any active user. We associate to Algorithm 13 a bipartite graph $\mathsf{G}^{\mathsf{ALT}}$ of active users and active paths where edges are drawn between users and their assigned paths, and have weight n minus the rank of the path. Let us denote the maximum weight maximum size matching of this graph for Algorithm 13 by $\mathsf{M}(\mathsf{G}^{\mathsf{ALT}})$. Note that for the algorithm to have an α rank approximation, $\mathsf{M}(\mathsf{G}^{\mathsf{ALT}})$ should have a perfect matching of weight at

least $n * cn - \sum_{i=1}^{\alpha * cn} i$.

Levels of Paths. We associate with each path a *level*, which is an integer in $[0, \log(n/2)]$, defined as the log of the fraction of users who got assigned this path. That is, if a path p was assigned to $n/2^i$ users in the deterministic stage of the algorithm, then the level of path p is i.

For every path p at level j, let $U(p)$ and $A(p)$ respectively denote the sets of users and active users who were assigned p. Note that $|U(p)| \geq 2^{t-j}$. Then let p_{j-k} for all $k \in [j]$ denote the path from some higher level also assigned to any user who was assigned p. Also let P_{j+k} for $k \geq 1$ denote the set of paths from any lower level also assigned to some user who was assigned p.

To prove Theorem 7.2.1, we now prove a series of lemmas about Algorithm 13. The first shows that every user can be assigned a distinct path from OPT using the alternate routing assignment given by Algorithm 13.

Lemma 7.2.1. *For any routing instance and any subset of cn users for any $c > 0$, $|\mathsf{M}(\mathsf{G}^{\mathsf{ALT}})| = cn$.*

Proof. We will prove that a perfect matching can be obtained by using edges in $\mathsf{M}(\mathsf{G}^{\mathsf{ALT}})$ only from the paths assigned in the deterministic stage of the algorithm.

We show by induction on l that if there are l levels of paths assigned as per Algorithm 13, then iteratively matching the largest rank path available to users arranged arbitrarily ends in a perfect matching. If there are $l + 1$ levels, then consider the subsequence of the users that would get assigned some path from the last level according to our heuristic. Remove these users and all the active paths that are in the last level from $\mathsf{G}^{\mathsf{ALT}}$. Note that as we assign these paths before any other path, the number of paths removed is at most (hence equal to) the number of active users removed.

Now all the remaining active users must have had one of the removed paths assigned to them. Note that for any integer j, a path p on level j is assigned to at most 2^{t-j} users. Hence the removed paths were each assigned to at most 2 users, and one of them got matched. Therefore, we now have half the number of unmatched active users, and paths of at most l levels. The number of active users for each path on any level $j < l$ is also equal to the number of users that are given some path in $P_{j+(l-j)}$, hence at most half the original number of users. Thus this is an alternate paths assignment of l levels, which ends in a perfect matching by inductive hypothesis.

For the base case, if there is a single level, then there is 1 active path and 1 active user, which is a perfect matching. This completes the proof. QED.

For the next lemma, let the *active rank* of an active path be its index in the vector of active paths ordered by decreasing rank in OPT.

Lemma 7.2.2. *The expected rank of the path with active rank (cn) is at most $(1 + 2/e)cn$.*

Proof. Consider a path p at level j where $2^j \leq cn$. There are $n/2^j$ users in $U(p)$. The probability

that this path is not active is,

$$\begin{aligned}
\mu &= \binom{n - n/2^j}{cn} \Big/ \binom{n}{cn} \\
&= \frac{(n - n/2^j)!}{(n - n/2^j - cn)!} \frac{n!}{(n - cn)!} \\
&= \prod_{i=1}^{n/2^j} \frac{n - n/2^j - cn + i}{n - n/2^j + i} \\
&\leq \left(\frac{n - cn}{n}\right)^{n/2^j} \\
&= (1 - c)^{n/2^j} \leq e^{-cn/2^j} \leq \frac{1}{e}.
\end{aligned} \tag{7.1}$$

Hence, the expected number of active paths from paths of level j is $2^j \cdot (1 - \mu) \geq 2^j(1 - 1/e)$. In other words, at least $(1 - 1/e)cn$ of the first cn paths are active in expectation. The rank of the $(cn)^{th}$ active path is the rank of the $(cn/e)^{th}$ active path from those ranked above cn. As each path on level j has at least one path in its set P_{j+k}, there are at least $(2^j(1 - 1/e))$ active paths on level $j+1$ as well. For worst case, if all of these paths are the last ranked paths on level $j+1$, that is, the expected active paths on this level have ranks $\{2cn, 2cn - 1, 2cn - 2, \dots\}$ then the first of these has rank $(1 + 1/e)cn + 1$, and the $(cn/e)^{th}$ active path has rank $(1 + 2/e)cn$. QED.

Lemma 7.2.3. $\mathsf{G}^{\mathsf{ALT}}$ *matches every user with a path of rank at most* $(1 + 2/e)cn$ *in expectation.*

Proof. From a similar reasoning as the proof of Lemma 7.2.1, we can show by induction on level j that we get a perfect matching in expectation by assigning users active paths from level $\log(2cn)$ and higher. We use Lemma 7.2.2 to show there are enough active paths on each level in expectation to match to at least half the remaining active users. The optimal matching can only do better, hence we get the lemma. QED.

We now refer to the online stochastic matching results from [124]. They show that when a matching exists in expectation, we can get the following.

Lemma 7.2.4 ([124])**.** *A* $(1 - 1/e)$ *fraction of the* cn *users will get matched to paths as per the expected offline matching, if users arrive in arbitrary order and choose the greedily best available path.*

We use this to prove our main Theorem 7.2.1.

Proof of Theorem 7.2.1. From Lemmas 7.2.3 and 7.2.4, we can say that $(1 - 1/e)cn$ users get matched to one of the first $(1 + 2/e)cn$ paths. The remaining users must be matched to a path at some lower level. While the expected active users bound from equation (7.1) does not apply for

these levels, we know that there are at least $cn/e(1-1/e)$ expected active paths assigned to the remaining cn/e active users at level $\log(cn/e)$ from the same equation. As each of these also gets a path from the lower levels, this means that there are at least $t = cn/e(1-1/e)$ expected active paths in all the lower levels. t users get matched to some path from level $\log(2(1+2/e)cn)$, and the remaining from the next level, as $2t > cn/e$. Thus, all users get matched to a path of rank at most $4(1+2/e)cn$. QED.

We now prove fairness guarantees of Algorithm 13.

Theorem 7.2.2. *Algorithm 13 is $EF1$ and 2-envy free.*

Proof. We first show the algorithm is $EF1$. Consider any two users i and j, and denote the path assigned to user $k \in \{i, j\}$ from level u be p_u^k. Suppose the paths assigned on the first l levels are the same for both, i.e., $p_u^i = p_u^j$ for all $u \in [l]$, and without loss of generality, i has been assigned the smaller rank path at level $l+1$, i.e., $p_{l+1}^i < p_{l+1}^j$. Then for each path assigned to j, i has a better rank path assigned to them, that is, $p_v^i < p_v^j$ for all $v \geq l+1$. Also, for each path assigned to j, the rank of the path assigned to i at the next level is bigger, that is, $p_v^j < p_{v+1}^i$ for all $v \geq l+1$. Hence, the algorithm is $EF1$, as it is $EF1$ for any pair of users.

Next we show 2-envy freeness. We first show by induction that the paths computed by the iterated greedy algorithm after any k^{th} iteration have cost at most twice the cost of the best path after that iteration. This is trivially true after the first iteration. For any later k, let path p be the best path before the next is computed. As recomputing p as the next best path induces a cost at most twice the cost of p, the greedy algorithm will only find a path with envy ratio at most 2 to p. Next, as every path computed in a previous iteration except p was 2-envy free with respect to p, and the best path after the next iteration cannot have a smaller cost than p, it will be 2-envy free with respect to the best path after the $(k+1)^{st}$ iteration as well. If the new path is also p, then the new cost of p with one more user can only be twice its earlier cost. As all other users had higher costs, the new cost cannot be more than twice that of any other path. Hence the assignment is 2-envy free with respect to p as well.

To summarize, all the paths computed by the iterated greedy algorithm have an envy ratio at most 2 with each other. As Algorithm 13 guarantees that distinct rank paths can be assigned to each user, the congestion on any path in the assignment formed by the algorithm is not worse than its congestion in the greedy algorithm, hence the envy ratio guarantee is maintained. Hence Algorithm 13 is 2-envy free. QED.

Algorithm 14: Route Assignment Algorithm for Identical Users

Input : $\langle n, s, t, C \rangle$

Output: Alternate routing assignments that assigns at most $O\left(\frac{\log^2 n}{\xi^2}\right)$ paths to each user

Let $N := \lceil \log_{1+\xi}(n) \rceil$
Choose $m = \lceil (1+\xi)n \rceil$ paths using the iterative greedy algorithm, $p_1, p_2, \ldots, p_m$
for $j = 1 \ldots N$ **do**
    let $B_j := \left\{ p_1, p_2, \ldots, p_{\lceil (1+\xi)^j \rceil} \right\}$
for *user* $i = 1 \ldots n$ **do**
    $R_i := \emptyset$
    for $j = 1 \ldots N$ **do**
        Sample $\frac{4}{\xi}(2 \ln |B_j| + 2)$ paths from B_j uniformly at random and put them into R_i

7.3 ANALYZING USERS VIA RANDOMIZATION

Given the assignments (or recommendations), users pick the best route available greedily among the recommendations. In 8, we will show that, for any fixed permutation of users π, i^{th} user in order will always be assigned to a path of rank at most $(1 + O(\xi)) \cdot i$ with constant probability. It is easy to see that by repeating the algorithm a constant number of times, the probability can be made arbitrarily small without affecting the asymptotic number of recommendations per user.

Theorem 7.3.1. *For any permutation of users π, above algorithm will be a rank-$(1 + O(\xi))$ approximation with probability $\geq \frac{1}{3}$.*

Proof. Let X_i be the event that each user π_j, for all $j \leq i$, is assigned to a path of rank at most $(1 + O(\xi))j$. Note that the recommendations for i^{th} user, R_i, are independent of $X_1, \ldots, X_{i-1}$. Furthermore $\Pr[X_1] = 1$, since the shortest path p_1 is available to all users.

Now consider i^{th} user. Let's define $j := \lceil \log_{1+\xi}(i) \rceil + 1$. If we can show that user is assigned to one of the paths from B_j which has not been assigned to any of the previous users, then this means the rank of the path chosen by this user will be at most $(1+\xi)^3 i \leq (1+3\xi)i$. The number of paths available to i^{th} user is at least $|B_j| - (i-1) \geq (1+\xi)^{\lceil \log_{1+\xi}(i) \rceil + 1} - (i-1) \geq \xi \cdot i \geq |B_j| \frac{\xi}{(1+\xi)^2} \geq \frac{\xi}{4}|B_j|$.

Since the recommendations from B_j are uniformly distributed, the probability that i^{th} user is given a route which has not been assigned yet is at least:

$$\Pr[X_i | X_{i-1}] \geq 1 - (1 - \frac{\xi}{4})^{\frac{4}{\xi}(2 \ln |B_j| + 2)} \geq 1 - \frac{1}{e^2 |B_j|^2} \geq 1 - \frac{1}{e^2 i^2}.$$

Consequently,

$$\Pr[X_1 \wedge X_2 \wedge \ldots \wedge X_n] \geq \prod_{i=1} \left(1 - \frac{1}{e^2 i^2}\right) \geq 1 - \sum_{i=1} \frac{1}{e^2 i^2} \geq 1 - \frac{\pi^2}{6e^2} \geq \frac{1}{3}. \qquad \text{QED.}$$

Theorem 7.3.2. *Algorithm 14 is 2-EF.*

Proof. From the same reasoning as of the proof for the deterministic algorithm 13, we can say that the iterated greedy algorithm itself assigns paths such that no two users have an envy ratio of more than 2. Hence, as the alternate path assignment of Algorithm 14 also assigns paths computed in distinct iterations to each user, it does not affect this guarantee. QED.

7.4 EXPERIMENTS

7.4.1 Algorithms

We simulate a generalized version of Algorithm 13, as shown in Algorithm 15, for demands with non-identical source and/or destination requirements. Not that for this case, the iterated greedy algorithm will generate different paths depending on the ordering of the users. Ideally, we would like to assign alternates that give a good guarantee over all possible permutations of the users, and over all possible optimal assignments. It turns out that a few random permutations suffice. Our algorithm computes the optimal greedy path assignment 3 times for 3 different permutations of the users. It then considers all subsequences of paths of identical users from each set, and runs Algorithm 13 to generate an alternate path assignment for these. The final set of alternates assigned to each user is the union of paths assigned for each random permutation of the users.

Heuristic. For a large number of users, Algorithm 15 could be inefficient, hence we run the algorithm on a simplified graph obtained as follows. There are two intuitive reasons that help us obtain a good heuristic. First, an $s-t$ pair of nodes that is far away from another pair will not affect congestion values on the optimal paths between the other pair, hence we can compute the greedy paths of both pairs separately. Second, there cannot be too many desirable distinct paths between pairs of nodes where the source and destination nodes are close to each other, and when the number of users is high, the optimal assignment will recompute older paths frequently as optimal in future iterations. Hence, we perform the following pre-processing.

Graph Sparsification. We select a large number of $s - t$ pairs of nodes randomly, and run the iterated greedy algorithm to compute a fixed number of paths for each. The number of paths increases with the distance between the source and destination nodes. We collect the edges that lie on any of these paths, and run Algorithm 15 on the subgraph formed with these edges, and for a

random demand over these $s-t$ pairs. The exact details of this preprocessing are specified in the next section.

Algorithm 15: $(3\log \mathsf{n})$-ALTALGO for non-identical users

Input : $\langle (n_i, s_i, t_i)_{i\in[k]}, C\rangle$
Output: Alternate routing assignment that assigns at most $O(\log n)$ paths to each user

for *all* $i \in [k]$ **do**
  $2^{t_i} :=$ smallest power of 2 higher than n_i
$L \leftarrow$ list of $\sum_i 2^{t_i}$ users with 2^{t_i} users associated with pair $s_i - t_i$.
for $j \in [3]$ **do**
  $L_j \leftarrow$ arbitrary permutation of L
  Compute $\sum_i (2^{t_i}-1)$ **OPT paths**
  $OPT_j := \emptyset$
  $T := \{0\}^m$ `// initialize congestion of every edge` e `to` $T_e = 0$
  for $i \in [\sum_k (2^{t_k}-1)]$ **do**
    $(s_i, t_i) \leftarrow$ source-destination pair associated with user i
    Find shortest path P_i from s_i to t_i in G where each edge has weight $c_e(T_e)$
    Add P_i to OPT_j
    for *edge* e *in* P_i **do**
      $T_e \leftarrow T_e + 1$
  Assigning alternate routes:
  1. Deterministically assign $O(3\log n)$ **paths**
for $i \in [k]$ **do**
  for $j \in [3]$ **do**
    $OPT_j^i :=$ subsequence of OPT_j with demand $s_i - t_i$
    $R_j^i \leftarrow$ Alternate route assignment of OPT_j^i as in Algorithm 13
  $R^i \leftarrow \cup_j R_j^i$ `// alternate route assignment to users with demand` $s_i - t_i$
Return $\cup_i R_i$

7.4.2 Data

Graphs We extract the the San Francisco Bay Area graph from Open Street Maps [118] (including the whole region from Berkeley to San Jose). For each road and intersection, we extracted its distance, number of lanes and maximum speed allowed.

Given the road data, we construct our graphs as follows:

- We add one node per road in each available direction;

- We add a directed edge from the node representing one road segment to the other if the corresponding turn is permitted.
- The cost of each directed edge is calculated as the sum of costs for the starting road segment and the corresponding turn.

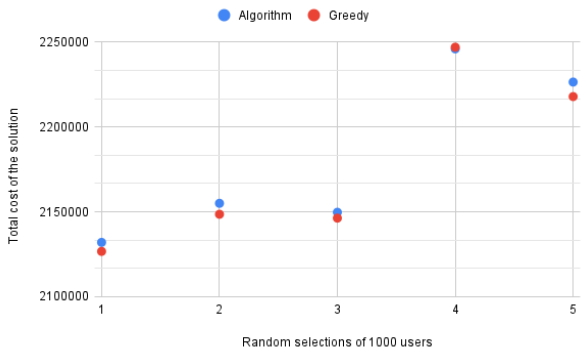
The costs per roads are calculated using the functional form of the Bureau of Public Roads [119], similar to [85, 120]. Specifically we set:

$$c_e(x) = \frac{t_e^f}{\kappa_e} x + t_e^f,$$

where t_e^f is the time needed to cross the edge when the road is empty, i.e., the free-flow travel time, and κ_e is the capacity of the street, defined as the number of lanes multiplied by the free-flow speed.

Parameters. To apply the graph sparsification heuristic, we select 9 sets of $s-t$ pairs of nodes uniformly at random from the original graph, where the i^{th} set has pairs of nodes that lie at distances $5i$ km to $5(i+1)$ km from each other. We compute $5(i+1)$ paths for each pair in the i^{th} set. We then select a demand flow for each $s-t$ pair as per the inverse square law between the demand and distance between the source and destination from [125]. Formally, the law is, $\rho(r, f) = \mu/(rf)^\nu$, where $\rho(r, f)$ is the flow between a pair of nodes, r is the distance between the nodes, f is the frequency of travel between the nodes, μ is a proportionality constant dependent on the area under observation, and $\nu \approx 2$. Using this law, we generate 3900 s-t pairs of nodes as follows and run the iterated greedy algorithm on each separately. The Bay area graph is sparsified by removing all edges that do not lie on any path of the above greedy computations.

s-t distance (r) (km)	# s-t pairs	# Paths / pair	# Users / pair (ρ)
5-10	1000	10	1000
10-15	675	15	675
15-20	500	20	500
20-25	400	25	400
25-30	350	30	350
30-35	300	35	300
35-40	250	40	250
40-45	225	45	225
45-50	200	50	200
Total	3900	91250	3900



(a) Comparing the total cost of an assignment of our algorithm with the greedy optimal over 5 iterations. The algorithm gives near-optimal costs for any selection.

(b) Envy ratio of the algorithm and the greedy optimal as a function of the distance between the source and destination nodes. For any distance range, our algorithm ensures approximate fairness.

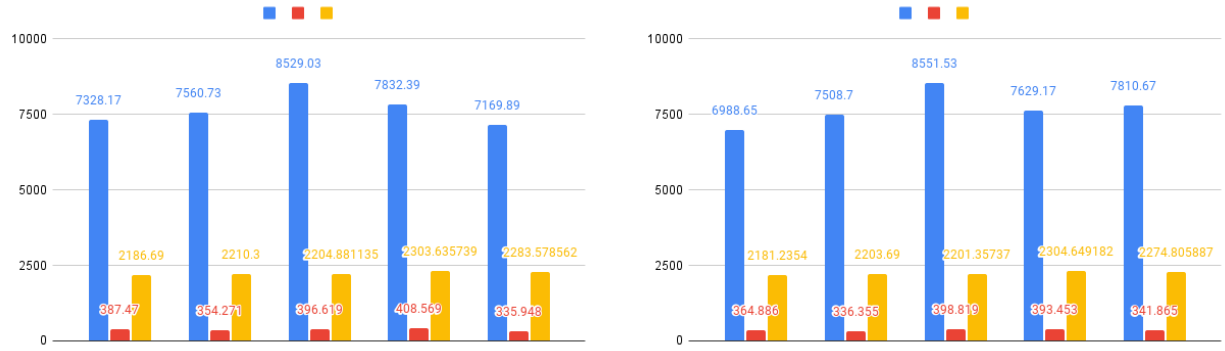
The resulting graph has $4.6M$ nodes and $2.4M$ edges. Out of these nodes, only $2.1M$ are non-isolated, and the average degree of the sparse graph is approximately 1.14. Compared to the original Bay Area graph which has $4.7M$ nodes and $10.1M$ edges and is well connected, this speeds up the path computation algorithm considerably. We also do not lose out on optimality as shown formally by comparing the congestion cost results of our algorithm with the optimal one.

We run Algorithm 15 on the sparse graph. We randomly select 1000 users for 5 iterations and generate comparison plots for analyzing the fairness and optimality of our algorithm.

7.4.3 Results

We illustrate the performance of our algorithm by comparing the fairness and optimality of the same with the optimal iterated greedy algorithm. For this, we take a random permutation of the selected users and run the optimal iterated greedy algorithm again. We then run the ROUTFrmALT corresponding to our alternate routing Algorithm 15 on this permutation of the users.

For both of these algorithms, we first compare the total congestion cost of the users obtained in the two algorithms in Figure 7.1a. We also show a comparison of the maximum, minimum and average cost of any user in each run of both algorithms. These plots are shown in Figures 7.2a and 7.2b. They show that the total cost of our algorithm does not reduce even after reducing the graph available for each user, that is, a union of alternate paths. Next we compare the fairness of both algorithms. Note that with non-identical users, the envy-ratios can be distinct for users of distinct $s-t$ pair set. We cumulate these by taking the maximum, minimum and average values over all distinct values. Figure 7.1b shows the envy-ratios obtained in both runs for each algorithm.



(a) Algorithm

(b) Greedy optimal

Figure 7.2: Plotting the maximum, minimum and average cost of both Algorithm 15 and the greedy optimal algorithm. The left-most bars mark the maximum cost, the middle bars the minimum and the right bars the average cost of each selection of users. It can be seen that for every iteration, according to any key statistical metric, the cost of our algorithm is always near-optimal to the cost of the greedy optimal solution

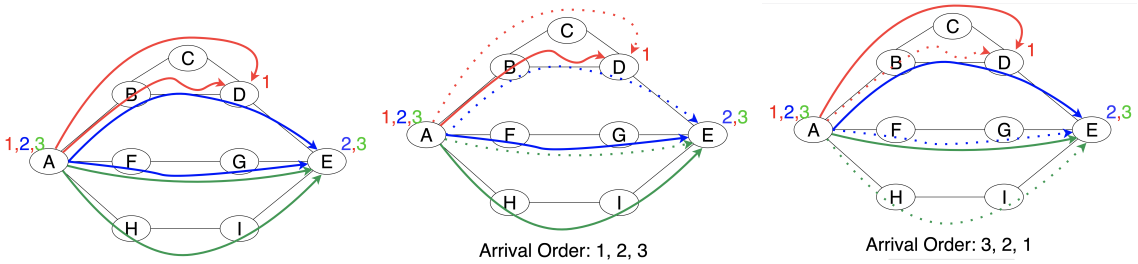


Figure 7.3: Illustration of our algorithm. There is one user who wants to go from A to D and two users who want to go from A to E. (Top) Suppose the first user is given routes (A, B, D) and (A, B, C, D); the second user is given routes (A, B, D, E) and (A, F, G, E) and the last user is given routes (A, F, G, E) and (A, H, I, E) all in order of preference. (Middle) If the users arrive in order 1, 2 and 3; then the second user will pick route (A, F, G, E) due to the congestion on edge (B, D); and the third user will pick the route (A, H, I, E). (Bottom) If the users arrive in opposite order; then the first user has to choose (A, B, C, D) due to the congestion on edge (B, D).

CHAPTER 8: CONCLUSION AND FUTURE WORK

This thesis focuses on computational aspects of the fair division problem with indivisible resources. We extend the theory to more general settings, namely agents with asymmetric entitlements/weights, mixed resources, and beyond additive valuations. Our work can be extended in several directions. We propose one specific problem for immediate future work, and describe some methods that can be used to solve it, followed by a broader fair division problem. We also propose two directions of research that are important, in our opinion, for fair division, but whose explorations will require several new ideas beyond this thesis.

SUBLINEAR FACTOR ALGORITHM FOR ASYMMETRIC NASH WELFARE WITH ADDITIVE VALUATIONS

The Nash welfare problem with asymmetric agents who have additive valuation functions, has two state-of-the-art results - the first is SMatch from this thesis (Chapter 3), which gives an $O(n)$ factor approximation, and the second is [38], who give an $O(\gamma^3)$ factor approximation, where n is the number of agents, and γ is the ratio of the largest and the smallest agent weights. The following are some ideas to improve these results to yield a sublinear factor algorithm.

Improving the analysis of SMatch for the symmetric agents sub-case. We simulated SMatch for the symmetric weights setting with general additive functions, and could not find an instance for which the algorithm gave a worse than 1.45 factor approximation. We also have shown a worst case analysis that confirms this factor for the restricted additive valuations special case. The first step would be to extend this restricted additive functions result to the general additive setting.

The current bottleneck in showing a worst case constant factor guarantee for the general additive setting, is our observation that every agent may lose at most n goods in every iteration of the repeated matching algorithm. This adds an $O(n)$ factor in the approximation factor. However, this analysis is naive. Note that not all the agents can lose $O(n)$ items in a single round, as only n items have been allocated in all. Further, the Nash welfare objective is, in some sense, computing the net loss to all the agents together. Hence, if only constantly many agents, or even a sublinear number of agents, lose $O(n)$ items in every round, the Nash welfare objective will not add an $O(n)$ factor in the final factor.

Ideas from submodular Nash welfare series of works. The Nash welfare problem with symmetric weight agents and submodular valuations has seen a remarkable series of improvements. Starting from our algorithm RepReMatch and its $O(n \log n)$ factor analysis, each work has intro-

duced new ideas that has resulted in a constant factor algorithm. A lot of these works use the idea of unmatching the higher valued items and rematching them, as a key subroutine. One direction would be to see if the ideas introduced by these works are useful in the asymmetric agents case.

SHARE BASED FAIRNESS FOR ASYMMETRIC AGENTS WITH BEYOND ADDITIVE VALUATIONS

Maximin share is a share based fairness notion. The idea of such notions is to compute a value or a share that is considered fair for each agent, and design algorithms that give every agent their fair share value. There are several proposed extensions of the maximin share for the asymmetric agents setting, but all of them have a linear factor worst case inapproximability. Another consideration is the beyond additive valuations setting, where the MMS problem has not been well-studied. Therefore, our two broad questions are,

- What share based fairness notions and problems should be explored for the fair division problem with asymmetric agents?
- Can one design constant factor algorithms, that improve known results, for MMS and other share based notions, when the agents have beyond-additive valuations?

APS. The first question's answer may be the any price share (APS) notion [21]. This is a new share based notion specifically proposed for the asymmetric agents setting. It also has the property that its value for the symmetric agents special case is equal or higher than the symmetric maximin share values. The introductory paper [21] show a constant factor approximation with asymmetric agents and additive valuations. Therefore, one possible extension could be similar approximations for beyond additive valuations. This may in turn be

MMS with beyond additive valuations. The MMS problem has been studied for the submodular valuations and beyond submodular valuation settings. To improve our understanding of MMS in these settings, one possibility is to relax the valuation functions and explore some of the rich classes that lie between additive and submodular functions. For instance, we propose the problems of MMS with weighted matroid rank valuations, and the more general Rado valuations.

AXIOMATIC APPROACH TO FAIR ROUTING

A broad direction is to develop an economic theory of fair routing. Classic fair division theory has evolved starting from designing axioms that fairness notions must satisfy, desirable additional

properties they can have, then proposing definitions that satisfy the axioms, and efficient methods to achieve them. Routing problems is a fundamental setting which should have their fairness theory developed in a similar way. To start this research, one can study the survey [126], that proposes axioms for fairness notions for the setting of goods with externalities, meaning settings where the utility of an agent is affected by the allocation to other agents as well. Congestion aware routing settings have externalities, as one agent's path affects the cost to other agents, when their paths intersect.

BEYOND WORST CASE HARDNESS OF FAIR DIVISION

Computational fair division has largely followed the classical notions of worst case analysis and traditional complexity theory. One may get a better understanding through exploring other notions. For instance the following two directions would be interesting to explore.

Smoothed and average case analysis. Most fairness notions are NP-hard, even for very simple settings, for instance even with 2 agents. Fair division theory has looked at relaxations like approximation and randomized algorithms as a solution to bypass the hardness. However, worst-case analysis may not necessarily be the right answer for understanding the hardness of fair division. Empirical works regularly show results along the lines of - for most instances, a particular notion exists, or has a good approximation algorithm, far better than what worst analysis has managed to prove. The paradigms of average case analysis and smoothed analysis of worst case inefficient algorithms may be better languages to explore intractability.

Proof complexity to improve understanding of existence. Proving the existence of fair allocations, for certain notions of fairness, is an important branch of fair division. For example, consider the problem of showing that an EFx allocation always exists. Significant efforts have been unable to resolve this question even when there are 4 agents with additive valuation functions. We wonder if this inability to prove such a statement is arising due to the complexity of such a proof - meaning, an instance that invalidates this statement is large, or the proof of existence is too complex. Proof complexity is a paradigm in complexity theory that explores such questions. The idea of this area is to systematically organize all types of proofs, creating an ordering of systems of proofs. Then for a chosen question, we show statements of the form that any system up to a certain rank in this ordering, would fail in providing a short proof of the question, thus partially indicating the hardness of the question. While proof complexity is largely developed to refute short proofs specifically of the classic SAT problem, we believe it is worthwhile to explore if there could be some kind of lifting

theorems or reductions, that extend the ideas to other problems, specifically fair division problems like the existence of EFx allocations.

REFERENCES

[1] J. Garg, P. Kulkarni, and R. Kulkarni, "Approximating nash social welfare under submodular valuations through (un) matchings," in *Proceedings of SODA*. SIAM, 2020, pp. 2673–2687.

[2] R. Kulkarni, R. Mehta, and S. Taki, "Indivisible mixed manna: On the computability of MMS + PO allocations," in *In Proceedings of EC*, 2021. [Online]. Available: https://arxiv.org/abs/2007.09133

[3] R. Kulkarni, P. Kulkarni, and R. Mehta, "Maximin share allocations for assignment valuations," in *In Proceedings of AAMAS*, 2023.

[4] E. Budish, "The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes," *Journal of Political Economy*, 2011.

[5] S. Gollapudi, K. Kollias, R. Kulkarni, and A. Sinop, "Fair algorithms for road traffic optimization," *Preprint available on request*, 2022.

[6] S. Gollapudi, K. Kollias, R. Kulkarni, and A. Sinop, "Semi-online algorithms for road traffic recommendations," *Preprint available on request*, 2022.

[7] H. Moulin, *Fair Division and Collective Welfare*. MIT Press, 2003.

[8] J. Robertson and W. Webb, *Cake-cutting algorithms: Be fair if you can*. CRC Press, 1998.

[9] T. P. Hill and K. E. Morrison, "Cutting cakes carefully," *The College Mathematics Journal*, vol. 41, no. 4, pp. 281–288, 2010.

[10] A. D. Procaccia, "Cake cutting: Not just child's play," *Communications of the ACM*, vol. 56, no. 7, pp. 78–87, 2013.

[11] H. Aziz, B. Li, H. Moulin, and X. Wu, "Algorithmic fair allocation of indivisible items: A survey and new questions," *ACM SIGecom Exchanges*, vol. 20, no. 1, pp. 24–40, 2022.

[12] G. Amanatidis, H. Aziz, G. Birmpas, A. Filos-Ratsikas, B. Li, H. Moulin, A. A. Voudouris, and X. Wu, "Fair division of indivisible goods: A survey," *arXiv preprint arXiv:2208.08782*, 2022.

[13] H. Steinhaus, "The problem of fair division," *Econometrica*, vol. 16, pp. 101–104, 1948.

[14] G. Gamov and M. Stern, "Puzzle-math," *Viking, New York*, 1958.

[15] H. R. Varian, "Equity, envy, and efficiency," *J. Econ. Theory*, vol. 9, no. 1, pp. 63 – 91, 1974.

[16] A. D. Procaccia and J. Wang, "Fair enough: Guaranteeing approximate maximin shares," in *EC*, 2014.

[17] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi, "On approximately fair allocations of indivisible goods," in *Proceedings of the 5th ACM Conference on Electronic Commerce*, 2004, pp. 125–131.

[18] J. Nash, "The bargaining problem," *Econometrica*, vol. 18, no. 2, pp. 155–162, 1950.

[19] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, "The unreasonable fairness of maximum nash welfare," *TEAC*, vol. 7, no. 3, pp. 1–32, 2019.

[20] J. Harsanyi and R. Selten, "A generalized Nash solution for two-person bargaining games with incomplete information," *Management Science*, vol. 18, pp. 80–106, 1972.

[21] M. Babaioff, T. Ezra, and U. Feige, "Fair-share allocations for agents with arbitrary entitlements," in *Proceedings of the 22nd ACM Conference on Economics and Computation*, 2021, pp. 127–127.

[22] M. Chakraborty, A. Igarashi, W. Suksompong, and Y. Zick, "Weighted envy-freeness in indivisible item allocation," *ACM Transactions on Economics and Computation (TEAC)*, vol. 9, no. 3, pp. 1–39, 2021.

[23] M. Chakraborty, E. Segal-Halevi, and W. Suksompong, "Weighted fairness notions for indivisible items revisited," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 4949–4956.

[24] N. Nisan, É. Tardos, T. Roughgarden, and V. Vazirani, Eds., *Algorithmic Game Theory*. Cambridge University Press, 2007.

[25] R. W. Rosenthal, "A class of games possessing pure-strategy nash equilibria," *International Journal of Game Theory*, vol. 2, pp. 65–67, 1973.

[26] J. Garg, M. Hoefer, and K. Mehlhorn, "Approximating the Nash social welfare with budget-additive valuations," in *SODA*, 2018.

[27] R. Cole and V. Gkatzelis, "Approximating the Nash social welfare with indivisible items," in *Symp. Theory of Computing (STOC)*, 2015, pp. 371–380.

[28] R. Cole, N. Devanur, V. Gkatzelis, K. Jain, T. Mai, V. Vazirani, and S. Yazdanbod, "Convex program duality, Fisher markets, and Nash social welfare," in *EC*, 2017.

[29] N. Anari, S. O. Gharan, A. Saberi, and M. Singh, "Nash Social Welfare, Matrix Permanent, and Stable Polynomials," in *8th ITCS*, 2017, pp. 1–12.

[30] S. Barman, S. K. Krishnamurthy, and R. Vaish, "Finding fair and efficient allocations," in *EC*, 2018.

[31] N. Anari, T. Mai, S. O. Gharan, and V. V. Vazirani, "Nash social welfare for indivisible items under separable, piecewise-linear concave utilities," in *29th Symp. Discrete Algorithms (SODA)*, 2018.

[32] Y. K. Cheung, B. Chaudhuri, J. Garg, N. Garg, M. Hoefer, and K. Mehlhorn, "On fair division of indivisible items," *CoRR*, vol. abs/1805.06232, 2018.

[33] S. Barman, U. Bhaskar, A. Krishna, and R. G. Sundaram, "Tight approximation algorithms for p-mean welfare under subadditive valuations," in *Proceedings of ESA*, 2020.

[34] B. R. Chaudhury, J. Garg, and R. Mehta, "Fair and efficient allocations under subadditive valuations," in *Proceedings of AAAI*, 2021.

[35] S. Dobzinski, N. Nisan, and M. Schapira, "Approximation algorithms for combinatorial auctions with complement-free bidders," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, 2005, pp. 610–618.

[36] S. Barman, A. Krishna, P. Kulkarni, and S. Narang, "Sublinear approximation algorithm for nash social welfare with xos valuations," *arXiv preprint arXiv:2110.00767*, 2021.

[37] W. Li and J. Vondrák, "Estimating the nash social welfare for coverage and other submodular valuations," in *Proceedings of SODA*. SIAM, 2021, pp. 1119–1130.

[38] J. Garg, E. Husic, and L. A. Vegh, "Approximating nash social welfare under rado valuations," in *Proceedings of STOC*, 2021.

[39] W. Li and J. Vondrák, "A constant-factor approximation algorithm for nash social welfare with submodular valuations," in *Proceedings of FOCS*. IEEE, 2022, pp. 25–36.

[40] J. Garg, E. Husić, W. Li, L. A. Végh, and J. Vondrák, "Approximating nash social welfare by matching and local search," *arXiv preprint arXiv:2211.03883*, 2022.

[41] T. T. Nguyen and J. Rothe, "Minimizing envy and maximizing average nash social welfare in the allocation of indivisible goods," *Discrete Applied Mathematics*, vol. 179, pp. 54–68, 2014.

[42] S. Barman, S. K. Krishnamurthy, and R. Vaish, "Greedy algorithms for maximizing nash social welfare," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 7–13.

[43] N.-T. Nguyen, T. T. Nguyen, M. Roos, and J. Rothe, "Computational complexity and approximability of social welfare optimization in multiagent resource allocation," *Autonomous agents and multi-agent systems*, vol. 28, no. 2, pp. 256–289, 2014.

[44] J. Garg, E. Husić, A. Murhekar, and L. Végh, "Tractable fragments of the maximum nash welfare problem," *arXiv preprint arXiv:2112.10199*, 2021.

[45] G. J. Woeginger, "A polynomial-time approximation scheme for maximizing the minimum machine completion time," *Operations Research Letters*, 1997.

[46] S. Bouveret and M. Lemaître, "Characterizing conflicts in fair division of indivisible goods using a scale of criteria," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 2, pp. 259–290, 2016.

[47] G. Amanatidis, E. Markakis, A. Nikzad, and A. Saberi, "Approximation algorithms for computing maximin share allocations," *ACM Transactions on Algorithms (TALG)*, vol. 13, no. 4, p. 52, 2017.

[48] S. Barman and S. K. Krishna Murthy, "Approximation algorithms for maximin fair division," in *Proceedings of the 2017 ACM Conference on Economics and Computation*. ACM, 2017, pp. 647–664.

[49] J. Garg, P. McGlaughlin, and S. Taki, "Approximating maximin share allocations," in *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[50] M. Ghodsi, M. Hajiaghayi, M. Seddighin, S. Seddighin, and H. Yami, "Fair allocation of indivisible goods: Improvements and generalizations," in *Proceedings of the 2018 ACM Conference on Economics and Computation*, ser. EC '18, 2018. [Online]. Available: https://doi.org/10.1145/3219166.3219238

[51] J. Garg and S. Taki, "An improved approximation algorithm for maximin shares," in *EC*, 2020. [Online]. Available: https://doi.org/10.1145/3391403.3399526

[52] A. Farhadi, M. Ghodsi, M. T. Hajiaghayi, S. Lahaie, D. M. Pennock, M. Seddighin, S. Seddighin, and H. Yami, "Fair allocation of indivisible goods to asymmetric agents," *J. Artif. Intell. Res.*, vol. 64, pp. 1–20, 2019.

[53] D. Kurokawa, A. D. Procaccia, and J. Wang, "When can the maximin share guarantee be guaranteed?" in *AAAI*, vol. 16, 2016, pp. 523–529.

[54] D. Kurokawa, A. D. Procaccia, and J. Wang, "Fair enough: Guaranteeing approximate maximin shares," *J. ACM*, vol. 65, no. 2, pp. 8:1–8:27, 2018.

[55] U. Feige, A. Sapir, and L. Tauber, "A tight negative example for mms fair allocations," *ArXiv*, vol. abs/2104.04977, 2021.

[56] L. Gourvès and J. Monnot, "On maximin share allocations in matroids," *Theor. Comput. Sci.*, vol. 754, pp. 50–64, 2019.

[57] R. Kulkarni, R. Mehta, and S. Taki, "On the PTAS for maximin shares in an indivisible mixed manna," in *Proceedings of AAAI*, vol. 35, 2021, pp. 5523–5530.

[58] H. Aziz, G. Rauchecker, G. Schryen, and T. Walsh, "Algorithms for max-min share fair allocation of indivisible chores," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[59] X. Huang and P. Lu, "An algorithmic framework for approximating maximin share allocation of chores," *arXiv preprint:1907.04505*, 2019.

[60] H. Aziz, B. Li, and X. Wu, "Strategyproof and approximately maxmin fair share allocation of chores," *arXiv preprint arXiv:1905.08925*, 2019.

[61] U. Feige and X. Huang, "On picking sequences for chores," *arXiv preprint arXiv:2211.13951*, 2022.

[62] S. Barman and S. K. Krishna Murthy, "Approximation algorithms for maximin fair division," in *Proceedings of the 2017 ACM Conference on Economics and Computation*, 2017, pp. 647–664.

[63] M. Seddighin and S. Seddighin, "Improved maximin guarantees for subadditive and fractionally subadditive fair allocation problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 5183–5190.

[64] S. Barman and P. Verma, "Truthful and fair mechanisms for matroid-rank valuations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 4801–4808.

[65] S. Barman, A. Biswas, S. K. Krishnamurthy, and Y. Narahari, "Groupwise maximin fair allocation of indivisible goods," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[66] B. R. Chaudhury, T. Kavitha, K. Mehlhorn, and A. Sgouritsa, "A little charity guarantees almost envy-freeness," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020, pp. 2658–2672.

[67] A. Biswas and S. Barman, "Fair division under cardinality constraints." in *IJCAI*, 2018, pp. 91–97.

[68] S. Brânzei, T. P. Michalak, T. Rahwan, K. Larson, and N. R. Jennings, "Matchings with externalities and attitudes," in *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13*, 2013.

[69] N. Anari, S. Ehsani, M. Ghodsi, N. Haghpanah, N. Immorlica, H. Mahini, and V. S. Mirrokni, "Equilibrium pricing with positive externalities," *Theor. Comput. Sci.*, vol. 476, pp. 1–15, 2013.

[70] X. Bei, A. Igarashi, X. Lu, and W. Suksompong, "Connected fair allocation of indivisible goods," *arXiv:1908.05433*, 2019.

[71] Z. Lonc and M. Truszczynski, "Maximin share allocations on cycles," *arXiv:1905.03038*, 2019.

[72] S. Barman, G. Ghalme, S. Jain, P. Kulkarni, and S. Narang, "Fair division of indivisible goods among strategic agents," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 1811–1813.

[73] H. Aziz, H. Chan, and B. Li, "Weighted maxmin fair share allocation of indivisible chores," *arXiv preprint arXiv:1906.07602*, 2019.

[74] H. Aziz, H. Chan, and B. Li, "Maxmin share fair allocation of indivisible chores to asymmetric agents," in *International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19*, 2019. [Online]. Available: http://dl.acm.org/citation.cfm?id=3331919 pp. 1787–1789.

[75] H. Aziz, P. Biró, J. Lang, J. Lesca, and J. Monnot, "Optimal reallocation under additive and ordinal preferences," in *International Conference on Autonomous Agents & Multiagent Systems*. ACM, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?id=2936984 pp. 402–410.

[76] J. Garg and P. McGlaughlin, "Improving nash social welfare approximations," in *IJCAI*, 2019.

[77] J. Garg and P. McGlaughlin, "Computing competitive equilibria with mixed manna," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 420–428.

[78] D. Chakrabarty, A. Mehta, and V. Nagarajan, "Fairness and optimality in congestion games," in *Proceedings of the 6th ACM Conference on Electronic Commerce*, 2005, pp. 52–57.

[79] J. R. Correa, A. S. Schulz, and N. E. S. Moses, "Fast, fair, and efficient flows in networks," *Oper. Res.*, vol. 55, no. 2, pp. 215–225, 2007.

[80] J. M. Kleinberg, Y. Rabani, and É. Tardos, "Fairness in routing and load balancing," *J. Comput. Syst. Sci.*, vol. 63, no. 1, pp. 2–20, 2001.

[81] C. A. Meyers and A. S. Schulz, "The complexity of welfare maximization in congestion games," *Networks*, vol. 59, no. 2, pp. 252–260, 2012.

[82] T. Roughgarden, "Stackelberg scheduling strategies," *SIAM J. Comput.*, vol. 33, no. 2, pp. 332–350, 2004.

[83] C. Swamy, "The effectiveness of stackelberg strategies and tolls for network congestion games," *ACM Trans. Algorithms*, vol. 8, no. 4, pp. 36:1–36:19, 2012.

[84] V. Bonifaci, T. Harks, and G. Schäfer, "Stackelberg routing in arbitrary networks," *Math. Oper. Res.*, vol. 35, no. 2, pp. 330–346, 2010.

[85] S. Colak, A. Lima, and M. Gonzalez, "Understanding congested travel in urban areas," *Nature Communications*, vol. 7, no. 10793, 2016.

[86] H. Youn, M. T. Gastner, and H. Jeong, "Price of anarchy in transportation networks: Efficiency and optimality control," *Phys. Rev. Lett.*, vol. 101, pp. 128–701, Sep 2008.

[87] K. Kollias, A. Chandrashekharapuram, L. Fawcett, S. Gollapudi, and A. K. Sinop, "Weighted stackelberg algorithms for road traffic optimization," in *29th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Beijing, China, November 2-5, 2021*. ACM, 2021.

[88] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[89] D. Eppstein, "Finding the k shortest paths," *SIAM J. Comput.*, vol. 28, no. 2, pp. 652–673, 1998. [Online]. Available: https://doi.org/10.1137/S0097539795290477

[90] M. Ben-Akiva, M. J. Bergman, A. J. Daly, and R. Ramaswamy, "Modeling inter-urban route choice behaviour," in *International Symposium on Transportation and Traffic Theory*. VNU Press, 1984, pp. 299–330.

[91] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck, "Alternative routes in road networks," *ACM J. Exp. Algorithmics*, vol. 18, Apr. 2013. [Online]. Available: https://doi.org/10.1145/2444016.2444019

[92] M. H. Kobitzsch, "Alternative route techniques and their applications to the stochastics on-time arrival problem," Ph.D. dissertation, Karlsruher Institut für Technologie (KIT), 2015.

[93] A. Paraskevopoulos and C. D. Zaroliagis, "Improved alternative route planning," in *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2013, September 5, 2013, Sophia Antipolis, France*, ser. OASICS, D. Frigioni and S. Stiller, Eds., vol. 33. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013, pp. 108–122.

[94] M. Kobitzsch, M. Radermacher, and D. Schieferdecker, "Evolution and evaluation of the penalty method for alternative graphs," in *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2013, September 5, 2013, Sophia Antipolis, France*, ser. OASICS, D. Frigioni and S. Stiller, Eds., vol. 33. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013, pp. 94–107.

[95] T. de la Barra and B. P. J. Anez, "Multidimensional path search and assignment," in *PTRC Summer Annual Meeting (SAM)*, 1993.

[96] D. Luxen and D. Schieferdecker, "Candidate sets for alternative routes in road networks," in *Experimental Algorithms - 11th International Symposium, SEA 2012, Bordeaux, France, June 7-9, 2012. Proceedings*, ser. Lecture Notes in Computer Science, R. Klasing, Ed., vol. 7276. Springer, 2012, pp. 260–270.

[97] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck, "Alternative routes in road networks," *ACM J. Exp. Algorithmics*, vol. 18, 2013.

[98] G. Aggarwal, S. Gollapudi, and A. K. Sinop, "Sketch-based algorithms for approximate shortest paths in road networks," in *Proceedings of the Web Conference 2021*, 2021, pp. 3918–3929.

[99] A. Sinop, L. Fawcett, S. Gollapudi, and K. Kollias, "Robust routing using electrical flows," in *SIGSPATIAL*, 2021.

[100] Z. Svitkina and L. Fleischer, "Submodular approximation: Sampling-based algorithms and lower bounds," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1715–1737, 2011.

[101] C. Chekuri, J. Vondrak, and R. Zenklusen, "Dependent randomized rounding via exchange properties of combinatorial structures," in *51st FOCS*. IEEE, 2010, pp. 575–584.

[102] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *40th Symp. Theory of Computing (STOC)*, 2008, pp. 67–74.

[103] S. Khot, R. Lipton, E. Markakis, and A. Mehta, "Inapproximability results for combinatorial auctions with submodular utility functions," *Algorithmica*, vol. 52, no. 1, pp. 3–18, 2008.

[104] B. Lehmann, D. Lehmann, and N. Nisan, "Combinatorial auctions with decreasing marginal utilities," *Games and Economic Behavior*, vol. 55, no. 2, pp. 270–296, 2006.

[105] V. Conitzer, R. Freeman, N. Shah, and J. W. Vaughan, "Group fairness for the allocation of indivisible goods," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[106] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi, "On approximately fair allocations of indivisible goods," in *Proceedings 5th ACM Conference on Electronic Commerce (EC-2004)*, 2004. [Online]. Available: https://doi.org/10.1145/988772.988792

[107] Y. T. Lee, Z. Song, and Q. Zhang, "Solving empirical risk minimization in the current matrix multiplication time," in *Conference on Learning Theory*, 2019, pp. 2140–2157.

[108] K. Jansen, K. Klein, and J. Verschae, "Closing the gap for makespan scheduling via sparsification techniques," in *43rd International Colloquium on Automata, Languages, and Programming, ICALP*, vol. 55, 2016, pp. 72:1–72:13.

[109] N. Nisan, "Bidding and allocation in combinatorial auctions," in *Proceedings of the 2nd ACM Conference on Electronic Commerce*, 2000, pp. 1–12.

[110] D. Buchfuhrer, S. Dughmi, H. Fu, R. Kleinberg, E. Mossel, C. Papadimitriou, M. Schapira, Y. Singer, and C. Umans, "Inapproximability for vcg-based combinatorial auctions," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2010, pp. 518–536.

[111] R. Day and S. Raghavan, "Assignment preferences and combinatorial auctions," in *University of Connecticut Working paper*. Citeseer, 2006.

[112] Y. Singer, "Budget feasible mechanisms," in *2010 IEEE 51st Annual Symposium on foundations of computer science*. IEEE, 2010, pp. 765–774.

[113] M. F. Balcan, F. Constantin, S. Iwata, and L. Wang, "Learning valuation functions," in *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 2012, pp. 4–1.

[114] S. Dobzinski, U. Feige, and M. Feldman, "Are gross substitutes a substitute for submodular valuations?" in *Proceedings of the 22nd ACM Conference on Economics and Computation*, 2021, pp. 390–408.

[115] M. Ostrovsky and R. Paes Leme, "Gross substitutes and endowed assignment valuations," *Theoretical Economics*, vol. 10, no. 3, pp. 853–865, 2015.

[116] R. P. Leme, "Gross substitutability: An algorithmic survey," *Games and Economic Behavior*, vol. 106, pp. 294–316, 2017.

[117] R. Bader, J. Dees, R. Geisberger, and P. Sanders, "Alternative route graphs in road networks," in *Theory and Practice of Algorithms in (Computer) Systems - First International ICST Conference, TAPAS 2011, Rome, Italy, April 18-20, 2011. Proceedings*, 2011, pp. 21–32.

[118] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org, 2017.

[119] B. of Public Roads, *Traffic assignment manual*. US Department of Commerce, 1964.

[120] B. Monnot, F. Benita, and G. Piliouras, "Routing games in the wild: Efficiency, equilibration and regret - large-scale field experiments in singapore," in *Web and Internet Economics - 13th International Conference, WINE 2017, Bangalore, India, December 17-20, 2017, Proceedings*, 2017, pp. 340–353.

[121] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *Experimental Algorithms, 7th International Workshop, WEA 2008, Provincetown, MA, USA, May 30-June 1, 2008, Proceedings*, ser. Lecture Notes in Computer Science, C. C. McGeoch, Ed., vol. 5038. Springer, 2008. [Online]. Available: https://doi.org/10.1007/978-3-540-68552-4_24 pp. 319–333.

[122] G. V. Batz, D. Delling, P. Sanders, and C. Vetter, "Time-dependent contraction hierarchies," in *Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments, ALENEX 2009, New York, New York, USA, January 3, 2009*, I. Finocchi and J. Hershberger, Eds. SIAM, 2009. [Online]. Available: https://doi.org/10.1137/1.9781611972894.10 pp. 97–105.

[123] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck, "Customizable route planning in road networks," *Transp. Sci.*, vol. 51, no. 2, pp. 566–591, 2017. [Online]. Available: https://doi.org/10.1287/trsc.2014.0579

[124] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan, "Online stochastic matching: Beating 1-1/e," in *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2009, pp. 117–126.

[125] M. Schläpfer, L. Dong, K. O'Keeffe, P. Santi, M. Szell, H. Salat, S. Anklesaria, M. Vazifeh, C. Ratti, and G. B. West, "The universal visitation law of human mobility," *Nature*, vol. 593, no. 7860, pp. 522–527, 2021.

[126] H. Moulin, "Uniform externalities: Two axioms for fair allocation," *Journal of Public Economics*, vol. 43, no. 3, pp. 305–326, 1990.