Algorithms for Achieving Fairness and Efficiency in Matching Problems

A THESIS

SUBMITTED FOR THE DEGREE OF

Doctor of Philosophy

IN THE

Faculty of Engineering

BY

Shivika Narang

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

June, 2023

Declaration of Originality

I, **Shivika Narang**, with SR No. **04-04-00-14-12-18-1-16435** hereby declare that the material presented in the thesis titled

Algorithms for Achieving Fairness and Efficiency in Matching Problems

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2018-2023**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Advisor Signature

DEDICATED TO

All the Giants

on Whose Shoulders I Have Stood and Seen Far

Acknowledgements

I am most grateful to my wonderful advisor Prof Y Narahari. Without his support and guidance throughout my masters and PhD, none of my research accomplishments would have been possible. Despite his many other commitments he always made time for me and allowed me to work on topics of my interest. He was unfailingly supportive and understanding and never added any unnecessary pressure on me. He was always very encouraging of my other collaborations, which I greatly appreciate. He also helped resolve many non-academic issues which made my life considerably easier.

I am also extremely grateful to my other faculty mentors. People are often hard pressed to find one kind and supportive faculty to guide them through their PhD, but I have been blessed with four. The first of these must be Dr Siddharth Barman. He has been an excellent mentor, faculty and collaborator throughout my PhD and even during my masters. He introduced me to the area of algorithmic fairness and I will be forever grateful for it. I have also been extremely fortunate to work with Prof Ioannis Caragiannis. While we were unable to meet in person so far due to Covid, he has been a wonderful collaborator and mentor to me over the last two years. I have been blessed to also be a part of Dr Hadi Hosseini's group, since August 2022, who also very generously hosted me at Pennsylvania State University. It was a pleasure working with him and visiting him during the four months I was at Penn State.

It has been a treat to have been able to work with all my other amazing collaborators. I am extremely grateful for Arpita Biswas, for braving a new area for both of us with me and for commiserating with me during the toughest time during my PhD. Pooja is by far my favourite person to bounce ideas off of and is great at grounding me when I jump too far. Anand has been the most patient of all my collaborators, especially to have joined the intimidating duo that Pooja and I make (so I'm told). Sanjukta and Tomasz have been fantastic collaborators and company since we met at Penn State. I will always be grateful to them for bringing in a complementary set of talents, making our combined research output soared.

None of my accomplishments, research or otherwise, would have been possible without my parents and my wonderful Nani. My incredible grandmother was the one who taught

me that I need to have faith in myself. Without her, I cannot imagine how my parents and I would have survived. My amazing parents gave their unending support, never once trying to force me to pursue any specific path. My mother taught by example that a PhD was definitely something one could do. It was seeing her complete a PhD alongside raising a teenage daughter and a full time job that gave me the confidence to pursue my own PhD. My father has been the most incredible father, by actually being there for me. Whether it was waking up early to pack my school lunches, or coming home and first checking on how my day was. He has gone beyond any other father I saw, actually wanting to get to know me as a person, rather than imposing his views and rules.

My incredible friends have been the biggest source of support for me throughout my life and especially during this PhD. Sanchi, who has been like a long lost twin, being the most supportive friend, even when she couldn't understand or relate to my problems. Pooja and Urvashi have been my constant supports since Masters. We have survived literal disaster, emotional disaster and even disasters due to typos in assignment questions together. Pooja has especially been my biggest support. My poor wrist wouldn't have healed without her toil. All my close friends through school and college have also been just as amazing. Avani, Soumya, Megha, Vishakha, Alka, Rashmi, Revati, Varsha and Avarna all of you have made my life better by being in it. My dopamine boosters Jinki, Jonghyun, Kibum, Minho, Taemin oppa, Wonwoo, Soonyoung, Seungcheol, Jeonghan, Joshua, Jun, Jihoon, Seokmin, Mingyu, Myungho, Seungkwan, Vernon and Chan. You all have pulled me out of many tough spots. Without all of you, I am not sure how I would have managed.

I have been blessed with incredible labmates. I am especially thankful to Ganesh for introducing me to matchings. I am extremely grateful to Shraddha, Pooja, Anand and Vishakha for proofreading this thesis. All my other amazing labmates and partners in crying Shweta, Arpita, Nidhi, Swapnil, Vishakha, Amal and Anand and the wonderful ones from Penn State: Sanjukta, Tomasz, Medha, Agha, Irfan, Deokjin. Since 2020, I have been supported by a Tata Consultancy Services fellowship which has been of great help, going well beyond the generous financial contribution.

I would also like to thank some people who may have simply been doing their jobs, but they have made my life infinitely better. Dr Ashwini Kulkarni and Dr Priyanka Lele, thank you for your guidance for navigating the stormy waters of grad school and life in general. Finally, I am very grateful to the staff at the CSA office for always kindly dealing with all my crises.

Isaac Newton said that he can see far because he stands on the shoulders of giants. You are all my giants and I have been standing on your shoulders.

Abstract

Matching problems arise in numerous practical settings. Fairness and efficiency are two desirable properties in most such real world scenarios. This dissertation work presents new approaches and models for capturing and solving fairness issues in different practical settings and develops algorithms to identify fair and/or efficient matchings. The thesis is organised into two logical parts: one-sided preferences and two-sided preferences.

Part 1: One-Sided Preferences

Fair and Efficient Delivery

Motivated by the classical delivery problem, we introduce a novel model of fair division where delivery tasks must be fairly distributed among a set of agents. The delivery tasks are placed on the vertices of a given acyclic graph. The cost incurred by the agents is determined by the distance they travel from the hub where they start to service their assigned tasks. We study the existence of fair and efficient allocations of tasks to agents. We choose the fairness notions: EF1 and MMS and efficiency notions: Pareto optimality and Social optimality. We find that while all these notions can be satisfied independently, the only combination of fairness and efficiency that can always be guaranteed is MMS and PO. For the remaining combinations, we provide characterisations of the space of instances for which they can be achieved. We find that most of the relevant problems are NP-Hard. We provide an XP-algorithm which finds the different combinations of fairness and efficiency whenever they exist.

Repeated Matchings

We propose a novel repeated matching model where the valuations of agents may change with how often they have received an item in the past. We study achieving fairness and efficiency separately as well as in conjunctions in this setting. We find that optimizing for social welfare is NP-Hard for general valuations and tractable when the valuations are monotone with time. We also prove that maximizing for social welfare over the space of EF1 repeated

matchings is NP-Hard. Further, we provide algorithms and non-existence results for EF1 and EFX repeated matchings in different settings.

Part 2: Two Sided Preferences

Fairness and Stability in Many-to-One Matchings

We seek to optimize a fairness measure over the space of stable many-to-one matchings, motivated by a college admissions setting. With leximin optimality as the fairness notion, we first show the intractability of this problem. We identify a minimal set of assumptions that makes this problem solvable in polynomial time. This requires that the agents on either side have the same ordinal rankings over the agents on the other side and that these must be strict. We show that on relaxing to weak rankings, the problem becomes APX-Hard. When we remove the ranking assumption but maintain strict preferences, the problem is NP-Hard. Additionally, we show that the leximin optimal stable matching can be efficiently computed in the special case of two colleges.

Incentive Compatibility in Stable Fractional Matchings

We investigate the existence of incentive compatible mechanisms that find stable fractional matchings. We show, for general settings, that no incentive compatible mechanism can be stable. We characterise the space of instances that have a unique stable fractional matching. We prove for this set of instances that any stable matching mechanism will be incentive compatible.

Publications and Manuscripts based on this Thesis

1. Shivika Narang and Y Narahari. A Study of Incentive Compatibility and Stability Issues in Fractional Matchings. Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2020).

1a. Shivika Narang and Y Narahari. On the Coexistence of Stability and Incentive Compatibility in Fractional Matchings. (Journal submission of above paper under review)

2. Shivika Narang, Arpita Biswas, Y Narahari. On Achieving Leximin Fairness and Stability in Many-to-One Matchings. Proceedings of the 21st International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2022).

2a. Shivika Narang, Arpita Biswas, Y Narahari. On Achieving Leximin Fairness and Stability in Many-to-One Matchings. (Journal submission of above paper under review)

3. Ioannis Caragiannis, Shivika Narang. Repeatedly Matching Agents to Items Fairly and Efficiently. To appear in Proceedings of the 22nd International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2023).

3a. Ioannis Caragiannis, Shivika Narang. Repeatedly Matching Agents to Items Fairly and Efficiently. (Journal submission of above paper under review)

4. Hadi Hosseini, Shivika Narang, Tomasz Was. Fair Distribution of Delivery Orders. (Conference submission under review)

Contents

List of Figures

LIST OF FIGURES

List of Tables

Notation	Definition
$\mathcal{A}$	Set of agents
$\mathcal{G}$	Set/graph of items
n	number of agents
m	number of items
v_i	valuation function of agent $i \in \mathcal{A}$
v	valuation function of all agents (identical valuations)
c	cost function of all agents $c = -v$
T	number of rounds
A	Allocation
A_i	Bundle of agent $i \in \mathcal{A}$ under allocation A
$SW(A)$	Social Welfare under allocation A
$X(\mathcal{G})$	Vertex Set of graph $\mathcal{G}$
$E(\mathcal{G})$	Edge set of graph $\mathcal{G}$

Notation used in Part 1

Notation	Definition
M	Set of men
W	Set of women
$\mathcal{S}$	Set of students
$\mathcal{C}$	Set/graph of items
n	number of agents in M, W, and $\mathcal{S}$
m	number of agents in $\mathcal{C}$
u_i	valuation function of m_i or s_i $i \in [n]$
v_j	valuation function of w_j or c_j $j \in [n]$
μ	Matching
$\mu(a)$	Partner(s) of a under μ

Notation used in Part 2

Chapter 1

Introduction

> This chapter presents an informal introduction to the space of matching problems and their applications, along with the pertinent fairness and efficiency issues that arise in solving the problems. We then provide an overview of the work presented in this thesis.

From the dawn of civilization there have been countless examples of people needing to be divided into pairs or groups for a variety of different purposes. There are just as many different instances of items to be divided amongst people. Be it matching students to colleges, tasks to workers or offices to employees, such matching settings have existed for millennia. The problem of coming up with adequate matchings in all these settings becomes even more challenging due to the personal preferences of the people involved. Thus, the need to ensure that these matchings are fair, i.e., they minimize the unhappiness of anyone, and efficient, or non-wasteful, is just as old as the matching problems.

(a) Labour Markets

(b) Ride Hailing Platforms

(c) Refugee Resettlement

Figure 1.1: Some Matching Applications

Most cultures have stories of people struggling to be just to those around them, against their own interests. Perhaps as a consequence of this, for the longest time, philosophers,

policy makers and ethicists have been pondering the exact meaning of fairness. They have recently been joined by computer scientists. With the use of automated systems becoming increasingly prevalent in solving such problems on a daily basis, it is crucial that the algorithms deployed in the backend are designed to meet these elusive ideals. This has led to the emergence of computational social choice being a significant part of the field of algorithmic game theory, at the intersection of Economics and Computer Science.

A simple example of fairness and efficiency issues in matchings can be seen when trying to distribute two chocolate bars between two children. Assuming that we correctly know the preferences of the children, let us now discuss how to divide the bars between them. If both bars are completely identical, each child can be given one bar each, and this solution would be fair and efficient. Now if both bars were equal in size but one bar had nuts and both children preferred having nuts in their chocolate, then half of each bar must be given to be completely fair. On the other hand, if one child preferred nuts, while the other was indifferent, giving half of each bar to each child would fair but not efficient. To be both fair and efficient the child who prefers nuts, should receive the bar with nuts and the other child the other bar. This increases the happiness of the child who prefers nuts without decreasing the happiness of the other child. Now if the bar with nuts were smaller in size, a fair and efficient solution would require more precise knowledge of exactly how much a bar with nuts is preferred to one without nuts. Thus, finding fair and efficient solutions can be very challenging, even in such seemingly simple settings.

The formal study of matchings based on agent preferences began with the seminal work of Gale and Shapley [52]. Since then extensive work has gone into the study of the theory and applications of matchings [1, 10, 33, 35, 82, 96, 101, 102, 104]. For their work in this area, David Gale and Alvin Roth won the Nobel Prize for Economics in 2012. This thesis is a continuation of the long line of work that tries to capture everyday situations mathematically as matching settings and attempts to develop algorithms that will produce fair and efficient solutions to these problems. In particular, this thesis looks at newer matching models which are either less explored or entirely unexplored by prior work. We explore the computational tractability of finding fair and efficient matchings in these new settings.

1.1 Matching Problems

Matchings can successfully model a large variety of everyday situations. Some of these include college admissions, hospital-resident matching, kidney exchange, refugee resettlement, ride-hailing platforms, labour markets, e-commerce markets, land division, and estate settlements among many others. Matchings can look very different across the various applications

and thus, the way fairness and efficiency manifest across the different matching models can vary. While there are many different matching models, we will only consider bipartite matching models, where there are two sets of agents or one set of agents and one of items which have to matched. By agents, we mean people who may have diverse preferences over the set of outcomes. By items, we mean things/people who do not have any preference over any of the outcomes. When agents have to be matched to agents, we say that the matching setting has two-sided preferences. When agents must be matched to items, we say that such settings have one-sided preferences.

Matchings with One-Sided Preferences

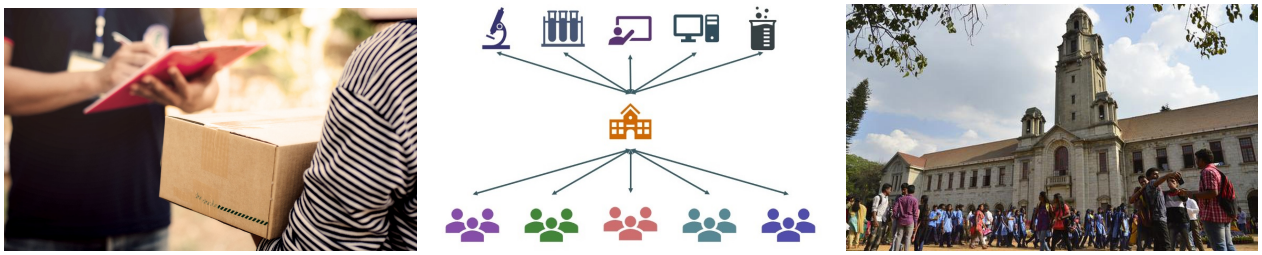
Agents may need to be matched to items for numerous reasons. Take for example, a simple situation of dividing delivery packages among the delivery agents. The packages have to be delivered to different locations across a town or city. It would typically be easier for a delivery agent to deliver a set of packages whose destinations are closer to each other than delivering a set of packages in different parts of the city. Thus an agent's preferences over the different sets of packages assigned to them would depend on the addresses that they need to be delivered to and the distances between them.

For the majority of the agents, the motivations behind their preferences would be consistent: the distance they need to travel, the weights of the packages, the total number of packages they are delivering etc. It would be unusual to have agents who prefer delivering one package over another, when the two packages have the same size and delivery location. Consequently, fairness and efficiency can be sought without requiring preferences to be elicited from every single delivery agent. Further, given that these conditions remain the same, the agents preferences will not change with time.

This need not extend to other one-sided matching settings. Consider a setting where research groups in a university must get access to research/experimental resources available at the university. A group working on game theory for example would have little use for time on a telescope. Even within computing facilities, different research goals would have diverse requirements which would be very customized to the projects. As a result, the preferences of different research groups could be very different and would require explicit elicitation. Further, these preferences could change with time. When first using a certain equipment, the researchers from a given group may spend a lot of time learning to use it and not be able to get any significant research results. With more time spent on the machine, their research output would gradually increase. Once the required experiments are run on a particular resource, the research group's value for it may decrease dramatically. Hence, if the schedules

of access are to be decided several weeks in advance, such variations must be taken into account.

Observe that in the examples given above the matching settings are different. Matching delivery agents to their packages requires that each package be the responsibility of exactly one agent. However, one agent may be allocated multiple packages to deliver within a day. Such a matching is called a many-to-one matching. For the case of scheduling the access to research equipment, it is reasonable to expect a researcher to work on exactly one project for the duration of an access slot of a few hours or so. Similarly, for most research equipment, typically, only one experiment can be run at a time. Thus, within one time slot, each researcher can be matched to only one facility and vice versa. Such a matching is called a one-one matching. However, over a month or so, it is feasible to grant different researchers access to a piece of equipment. Analogously, a researcher may get time on different pieces of equipment/resources over this time. Thus, over the scope of the time for which a schedule is being made, this would be a many-to-many matching. These differences in matching requirements persist even when agents have to be matched to agents.



(a) Package Delivery (b) Resource Matching (c) College Admissions

Figure 1.2: More Examples of Matching Settings

Two-Sided Preferences

The simplest and perhaps the most well studied example of many-to-one matchings with two-sided preferences comes from school choice or college admissions. Here students must be matched to colleges. Typically, a student would only attend one college (for the duration of a program at least) but a college admits many students to any one of its programs. Here too, students can have varied preferences over the colleges. Colleges in turn can have differing preferences over the students. These variations can come from differences in the programs that are offered at the various colleges, and the students inclinations. They can also come from a variety of other factors such as different methods for estimating the quality of a student/college, location preferences, financial constraints etc.

There can often be homogeneity in agent preferences in these settings however. In India for example, there are centralized entrance exams for admissions into specific programs, which become the basis for admission to the large majority of colleges offering those programs. For these colleges, the students are ranked based on their scores to these entrance exams. For students, their knowledge of the various colleges offering the program they're interested would be informed by various rankings and surveys produced by newspapers and magazines. As a result, there can be a lot of consistency in preferences. Such consistency can be useful for designing algorithms. We find that such consistency helps us not only in many-to-one matchings, but also in the case of two-sided one-one matchings. This type of consistency helps us find matchings that satisfy efficiency and fairness notions.

Agent Preferences

However we choose to efficiency and fairness in the various matching settings, it is necessary to take into account the agents' preferences over various matching outcomes. For the scope of this thesis, we shall assume that agents are indifferent over any outcomes in which their own matching is the same. For all remaining matchings, agent preferences can be expressed either by an ordinal relation over the outcomes, called ordinal valuations, or by a cardinal function, called the valuation function of the agent.

When preferences are expressed ordinally, we will assume a single linear order over all matching outcomes. When preferences are expressed cardinally, for each agent, every matching outcome will be associated with a cardinal value, the value of the agent for that outcome. An agent will prefer one outcome to another if their value is higher for it. Thus, given cardinal valuations, we can construct the ordinal valuations that are implied. Finally, we shall assume that all agents preferences are expressed the same way, whether ordinally or cardinally. We can now discuss how efficiency and fairness can be defined.

1.2 Efficiency in Matchings

Efficiency is a somewhat overloaded term in economics and computation. Algorithmic efficiency typically refers to the amount of time an algorithm takes as a function of the size of its input. Economic efficiency, on the other hand, looks at efficiency from the point of view of the agents involved, specifically their preferences. While we shall be discussing the time complexity of our algorithms, by efficiency, we shall be referring to a dominantly economic viewpoint of efficiency.

There can be many different ways to measure the efficiency of a matching outcome. Clearly, any such definition must take into account the preferences or valuations of the agents

involved. The most basic way is using *Pareto Optimality* (PO). Informally, a matching is said to be Pareto Optimal if there does not exist another matching which gives an agent higher value, without reducing the value obtained by another agent. While it is not trivial to satisfy, or even trivial to verify if a given matching is PO, it is not hard to see that this leaves a room for very many different matchings. Take for instance, a simple many-to-one matching setting n identical agents and m identical items. Any agent's value for the matched set (of items), is simply the number of items matched to them. Here, any matching would be Pareto Optimal.

Consequently, stricter notions of efficiency are often pursued. A particularly popular one is *Social Welfare*, which aims to maximize the sum of all the agents' values. Clearly, this requires that agents express their preferences as cardinal valuations. However, the majority of the work done on two-sided matchings considers settings with ordinal valuations. While Pareto Optimality is well defined here, social welfare is not. As a result in these settings, PO is strengthened to *stability*: there should not be any two agents who would like to abandon their respective partners and match with each other. Stability was first introduced by Gale and Shapley [52], and has since become a standard requirement for most work on two-sided matchings. Even with cardinal valuations, being stable is a standard goal for two-sided matchings. It is simple to see that stability becomes somewhat trivial in matching settings with one-sided preferences.

While matchings that are stable or those that maximize social welfare will always be Pareto optimal, they need not imply each other. That is, a social welfare maximizing matching need not be stable and vice versa. Matchings that optimize for these efficiency measures need not be fair themselves. Recall the example of dividing m identical items among n identical agents. Giving one single agent *all* of the items will satisfy Pareto Optimality and also maximize social welfare. This is clearly unfair. As a result, it is important to specifically seek fairness in the matching outcomes.

1.3 Fairness in Matchings

Like efficiency, there are various approaches towards fairness [5, 9, 30, 86, 92]. One is an egalitarian or Rawlsian approach which aims to maximize the happiness of the least happy person. More formally, the aim is to maximize the minimum value attained by any of the agents. This problem is also called the Santa Claus problem. The notion of leximin optimality takes this approach one step further. Over all matchings, it seeks to maximize the minimum value, and if there are multiple such matchings, *out of these* it seeks the one that maximizes the second lowest value. This continues till either there is only one matching left or all possible values are optimized for, contingent on the previous ones. We find that this notion is

able to capture fairness quite well for both one-sided and two-sided preferences.

The problem with this approach to fairness is that it is not verifiable by the agents involved, without knowledge of the preferences of other agents. A similar but verifiable approach to egalitarian welfare, specifically for many-to-one matchings one-sided preferences, is maximin fairness. All the agents are asked to partition the items into n disjoint sets (where n is the number of agents). The matching ultimately chosen need not be the one proposed by the agent, but will ensure that the agent gets at least as much value as the minimum out of all their values for the n sets specified. Consequently, agents would then themselves try to maximize the minimum value they would receive out of any of the n sets. This is called the maximin fair share. A matching that gives each agent value at least as much as their maximin fair share is said to satisfy maximin fairness (MMS).

It is worth noting that leximin optimality is an optimization based fairness notion whereas maximin fairness is a threshold based fairness notion. Consequently, a leximin optimal matching always exists but the existence of an MMS matching is not trivial to establish. In fact, an MMS matching need not always exist with diverse valuations. However, when agents have identical valuations, it can be shown that leximin optimality implies maximin fairness. It is not very surprising though that trying to attain either is computationally intractable in general.

An alternate verifiable approach to fairness is through *envy-freeness*. A matching is said to be envy-free, if no agent prefers the matching of another agent to their own. This is relatively straightforward for agents to verify on their own, with only knowledge of their own valuations and the whole matching required. Unfortunately, envy-free solutions need not exist when matchings are forced to be integral, i.e. when agents and items can't be partially matched. As a result, in such settings, we study *envy-freeness up to one item/agent*(EF1), where an agent may envy the matching of another agent, but this envy would be mitigated should one item or agent be removed from the other's matching. While all one-one matchings are EF1, for most typically studied many-to-one matching settings, EF1 matchings can usually be found in polynomial time. Further, the EF1 property is also verifiable by agents. Consequently, it has become a rather popular fairness notion among the computational social choice community.

1.4 Research Directions

There has been significant work on studying fairness and efficiency in matching settings. The work in this space has mostly focused on achieving fairness or efficiency under settings by a centralized process, where agents report their preferences and based on them a matching is selected. The study of one-sided and two-sided preferences has been largely independent

in the past. Work on two-sided preferences has often focused on stable matchings, looking at the various properties of the space of stable matchings [103, 114] or those of matching mechanisms that produce stable matchings [2, 66, 101, 115, 119]. The large majority of this work has assumed that agents have ordinal valuations. Consequently, fairness considerations have either been procedural [6, 74] or have tried to minimize the worst rank matched to any agent [68, 114]. Cardinal valuations in two-sided matching settings are mostly recent [36, 49, 59, 71, 90, 91] and influenced by the work done on fairness in one-sided settings.

One-sided matching settings, typically called allocation settings, have largely considered cardinal valuations. Efficiency investigations have largely focused on social welfare [17, 21, 44]. Fairness in allocation settings has considered Envy-freeness and its relaxations [14, 17, 33, 37, 41, 82, 96] as well as exact and approximate MMS [10, 12, 14, 33, 63, 65]. Leximin optimality has been studied, usually to show the existence of other fairness notions or for restricted settings [21, 29, 41, 45, 77, 96]. Several other fairness concepts have been also been studied in these settings such as Nash Social Welfare, Equitability and its relaxations. These notions of fairness have been studied both independently and in conjunction with the ones we study in this thesis. Some work in this space has also looked at algorithms that satisfy both fairness and efficiency.

Co-existence of Fairness and Efficiency

While it is crucial to study both fairness and efficiency independently, their co-existence is extremely desirable. Analogous to the case of efficient matchings, it is not necessary that an arbitrary matching that satisfies EF1 or MMS will always satisfy Pareto Optimality. There have been many attempts to find "welfare" functions that satisfy both fairness and efficiency. To this end the Pigou-Dalton principle was proposed. It states that any welfare function can only be optimized by solutions where a transfer of value from a richer agent to a poorer one results in a solution where the richer agent is now poorer than the poorer agent was initially. That is, all other things being equal, a welfare function should give higher value to solutions where agent values are more equitable.

Clearly this is satisfied by Leximin Optimality. Additionally, leximin also satisfies Pareto Optimality. Another fairness notion that satisfies the Pigou-Dalton principle is Nash Social Welfare [18, 19, 35]. A matching that maximizes Nash Social Welfare will also be PO. Prior Work which considers fairness and efficiency simultaneously has also looked at both the existence and computation of fair allocations that satisfy Pareto optimality [15, 35]. There has also been some work on maximizing social welfare over the space of fair matchings [17, 21].

The fairness and efficiency criteria described, with the exception of stability, can only be

formally defined when agents' preferences can be captured by cardinal valuation functions. As we look at fairness in terms of envy-freeness and its relaxations and leximin optimality, this dissertation only considers settings where agent preferences are cardinally expressed.

1.5 Thesis Overview

In this thesis, we aim to achieve fairness and efficiency in different matching settings. We provide an overview of the settings considered in each chapter in Figure 1.3. The dissertation is divided into two logical parts: one-sided preferences and two-sided preferences. In each we consider one setting with many-to-one matchings and one with one-one matchings. For each setting studied, we aim to resolve whether polynomial time algorithms can find fair and/or efficient matchings. Before discussing our technical contributions, we discuss necessary preliminaries in Chapter 2. Here, we discuss the necessary definitions of matchings, efficiency and fairness. We also provide a history of the classical results in these settings.

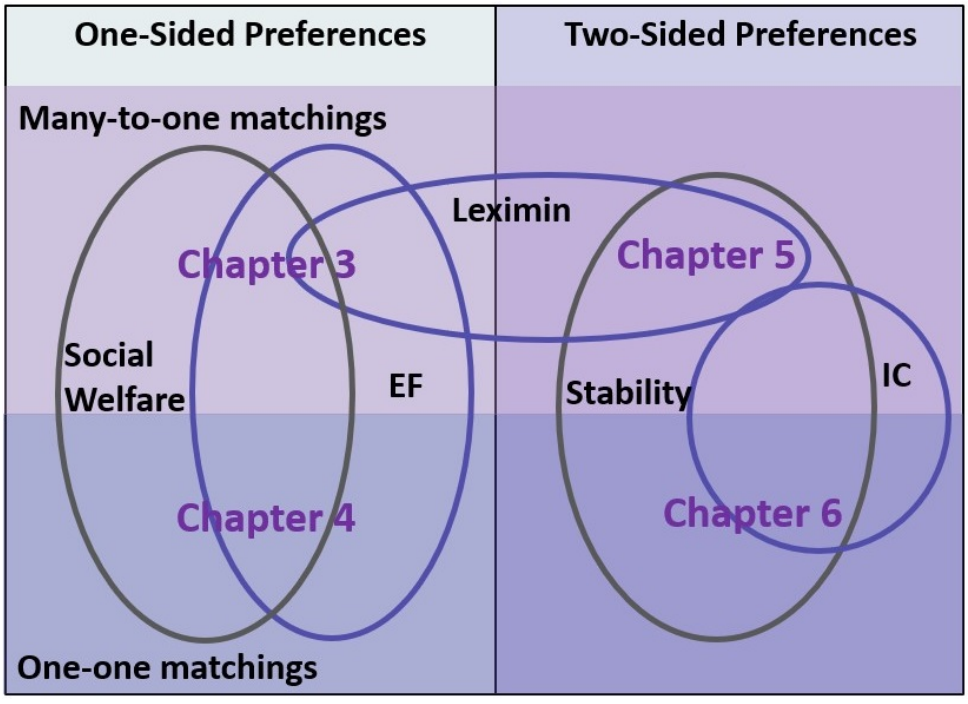


Figure 1.3: Thesis Overview

1.5.1 Part 1: One-Sided Preferences

One-sided preferences have been widely studied, largely for allocation settings where agents may get multiple items. Within this, a variety of different fairness notions have been studied. Most basic settings assume that agents values for a set of items is the sum of their values for

each item. Some prior work has also looked at more complicated valuations that need not be additive, but may be submodular or even subadditive. Further, these values do not typically change with time. In this part, we propose two new realistic models where fair and efficient matchings are needed. For both we look at the existence and computation of efficient and/or fair matchings. We find that it is computationally intractable to find matchings that satisfy both fairness and efficiency.

Chapter 3: Fair and Efficient Matching of Delivery Tasks to Agents

This chapter looks at delivery settings where agent valuations are typically identical. Here, the values are all non-positive. That is, agents have a cost of servicing their assigned orders. These costs depend on how much the agents have to travel in order to delivery the packages assigned to them. In this setting, we assume that there are n agents and an unweighted graph $\mathcal{G} = (X, E)$ with a special node h also called the hub. Each node is the delivery location of one package. The cost of an agent is the number of edges that the agent has to travel to start from the hub, service all the orders assigned and return to the hub. As finding an optimal route to service the assigned orders is clearly computationally intractable, we assume that the given graph is a tree. This makes the routing problem as well as finding social welfare maximizing solutions, straightforward.

We consider both EF1 and MMS in this space. We find that both EF1 and MMS solutions always exist (Section 3.3). However, fair solutions that maximize social welfare need not (Section 3.4). We also consider Pareto optimality and find that while EF1 and PO need not co-exist, MMS and PO do, via leximin. Further, with the exception of EF1 solutions that need not satisfy any efficiency guarantee, finding MMS solutions and all other combinations of fairness and/or efficiency is NP-Hard. We then develop an exponential-time algorithm, parameterized in the number of agents, that finds the Pareto frontier, and can be used to find any combination of fairness and efficiency in that we consider (Section 3.5). We also complement these theoretical results with detailed experimentation (Section 3.6).

Chapter 4: Repeatedly Matching Items to Agents Fairly and Efficiently

We introduce a new model of repeated matchings where the value of an agent for an item depends on how often they have been matched in the past. The same set of n agents and n items have to be repeatedly matched over T rounds. This makes problems that may be straightforward for one-shot matchings harder. We find that maximizing social welfare in this setting is NP-Hard, even when $T = 3$ (Section 4.3). However, this problem becomes tractable when the valuations are monotone in time. This is true for both monotone increasing and monotone decreasing values (Section 4.3.2).

For fairness, we consider relaxations of envy-freeness. We find that EF1 repeated matchings need not exist when item values can be positive or negative (Section 4.4). For the case of identical valuations and even the case of time-constant valuations, EF1 repeated matchings always exist and can be found tractably. Thus, for the case of time-constant valuations fair (EF1) and efficient (Social welfare maximizing) matchings can be found in polynomial time. However, maximizing social welfare over EF1 repeated matchings proves to APX-Hard (Section 4.5.1). We also give an algorithm for finding EF1 repeated matchings under general (non-negative) valuations for $T \bmod n \in \{0, \pm 1, 2\}$ We propose an alternate relaxation of envy-freeness called *swap envy-freeness* (swapEF) and show that it exists for mixed items for all the settings where EF1 repeated matchings exist for non-negative valuations (Section 4.6).

1.5.2 Part 2: Two-Sided Preferences

With two-sided matchings, stability is often non-negotiable. Consequently, for the space of two-sided matchings, we keep stability as an invariant. Here for one-one matchings, EF1 is satisfied by default. As a result, we consider fairness only in two-sided many-to-one matchings. Further, as these settings have been well studied in the past, especially with regards to stability, we can look at satisfying incentive compatibility in addition to stability.

Chapter 5: Achieving Fairness and Stability in Many-to-One Matchings

We consider fairness in stable many-to-one matchings where n students need to be matched to m colleges. While envy-freeness and its relaxations are not compatible with stability, we find that leximin optimality proves to be able to ensure fairness to agents on either side. We provide a comprehensive exploration of when the problem of finding the leximin optimal stable matching is computationally tractable. We find that finding the leximin optimal over the space of stable matchings is NP-Hard in general.

We then identify a minimal set of assumptions that makes this problem tractable. These are strict preferences and rankings. The combination of these two properties gives a structure over the space of stable matchings, which we use to iterate over this space and optimize for leximin (Section 5.3). For the settings where agents have strict rankings over agents on the other side we find an $O(m^2n^2)$ time algorithm. If agent values are also isometric, we can find the leximin optimal over all stable matchings in essentially linear time.

Relaxing either the requirement of rankings or strict preferences makes the problem computationally hard (Section 5.4). Finding the leximin optimal stable matching when the only assumption is isometric valuations is NP-Hard, even with only $m = 2$ colleges. Thus strict

rankings are essential for tractability. We find that relaxing strict rankings to weak rankings makes the problem APX-Hard. Meanwhile, removing the ranking requirement, but still insisting on strict preferences, the problem remains intractable. However, this reduction is from a problem that has an FPTAS, and we show that for $m = 2$ colleges and n students, the problem of finding the leximin optimal over stable matchings can be solved in time polynomial in n.

Chapter 6: Stability and Incentive Compatibility in Fractional Matchings

In the final technical chapter, we consider *fractional* one-one matchings. Fractional matchings are useful in modelling a large variety of everyday situations. The study of stable fractional matchings is relatively recent, and thus incentive compatibility concerns were hitherto unexplored. Incentive compatibility requires that agents have an incentive to be truthfully report their true preferences. In other words, an agent should not be able to receive a more preferable outcome by lying about their true preferences. Clearly, there are many real life matching applications where this would be an extremely desirable property.

We look at a setting with n men and n women for which we need to find stable fractional matchings. We find that in general, no matching mechanism that always returns a stable fractional matching can be incentive compatible (Section 6.3). In fact, if we were to relax the stability requirement to $1/2$, even then, incentive compatibility is not possible. We then find a space of matchings instances, which we call CMFP, where *all* stable matching mechanisms will be incentive compatible (Section 6.3.2). We show that under strict preferences, a matching instance has a *unique* stable fractional matching if and only if it belongs to this space. Further, testing for membership in this space can be done in $O(n^2)$ time. In order to characterize the space of matching instances with unique stable fractional matchings, we give an algorithm that returns a stable matching that is not integral whenever the instance is not in CMFP. Through this we also prove an interesting structural property of the space of stable fractional matchings.

We believe that this dissertation investigates several new, relevant and interesting problems in the space of applied matching theory. We believe this work will encourage further research in this space. We conclude in Chapter 7 with a summary of our contributions and directions for future work.

Chapter 2

Preliminaries

This chapter provides a summary of the relevant topics required to understand the technical details in this topics. We go over these topics to the extent required for this thesis. We start with some preliminaries and formally define matching. We then specifically look at one-sided matchings. We define the various notions of efficiency and give a brief overview of the work done in the space of stable matchings. We then define the necessary definitions of fairness for this thesis and give a brief history of classical results in both one-sided and two-sided matching settings.

2.1 Matching Definitions

The study of matchings covers a variety of practical contexts. The most commonly studied from these is one-to-one matchings, many-to-one matchings and many-to-many matchings. The large majority of these settings look at a bipartite matching setting with two sets of agents with preferences, based on which they must be matched. To this end, a typical matchings instance is described using two sets of agents, say $\mathcal{L}$ and $\mathcal{R}$, along with valuation functions capturing their preferences. We shall use u_i to denote the valuation function of agent $i \in \mathcal{L}$ and v_j to denote the valuation function of agent $i \in \mathcal{R}$. Thus, a matchings instance is typically denoted by a tuple $\langle \mathcal{L}, \mathcal{R}, (u_i)_{i\in\mathcal{L}}, (v_j)_{j\in\mathcal{R}} \rangle$.

For the purposes of this thesis, a matching μ is a subset of the edges in the complete bipartite graph between $\mathcal{L}$ and $\mathcal{R}$. Alternately, the notation $\mu(a)$ is used to denote the set of a's partners under μ. For any $a \in \mathcal{L} \cup \mathcal{R}$, the partner(s) of a under μ are the set of agents $\{a'$ s.t. $(a, a') \in \mu\}$. A one-to-one matching is one such that for any agent there is at most one incident edge in the matching. An example of such a setting is matching patients to hospital

beds in various facilities. Depending on the type of illness, patients have different values for the various types of beds, based on the facilities provided with each type of bed. The nature of care provided to the patient dictates the preferences of the hospital for having each patient in that type of bed. Other examples include matching employees to office spaces, matching research groups to research facilities and matching students to beds in college dorms.

A many-to-many matching typically allows an agent to have multiple incident edges in the matching. They are useful in capturing setting such as matching students to courses where multiple students may attend the same course and each student may attend multiple course. Additionally many-to-many matchings are often instrumental in modelling repeated one-one matchings where multiple repetitions/rounds of the matching are independent of past outcomes. This is exemplified by ride hailing platforms where both riders and drivers do multiple trips.

Finally, the third well-studied model of bipartite matchings is that of many-to-one matchings. A many-to-one matching has one set of agents, say $\mathcal{L}$ for whom at most one incident edge can be included in the matching, whereas the agents from $\mathcal{R}$ do not have this restriction. Such matchings are motivated most commonly as college admissions or hospital resident matchings. Many-to-one matchings, where *all* the agents in $\mathcal{L}$ are indifferent over the agents in $\mathcal{R}$ are typically called allocations. For such settings, preferences are elicited only from the agents in $\mathcal{R}$.

2.1.1 Valuations and Preferences

The set of partners of an agent shall determine their value for the matching. An agent's preferences over various matchings are determined by their preferences over their partners under the respective matchings. By an agent's preference, we shall mean an ordinal relation or a linear ordering over the other set of agents. When there are ties in this relation, such preferences are called weak preferences. When there are no ties, these preferences are strict.

The majority of work on two-sided matchings, does in fact only consider these ordinal preferences. However, ordinal preferences can be very restrictive in being able to capture the efficiency and fairness of matchings. Thus, for this thesis we use cardinal valuation functions to capture the preferences of the agents. This functions attach a value for all agents. The values of an agent are only specified for agents of the opposite set. From these valuation functions, we can define the value that the agent has for a given matching.

For most of this thesis, the value that an agent has for a matching is the sum of their values for all their partners under the matching. That is for $i \in \mathcal{L}$, the value for matching μ is $u_i(\mu) = \sum_{j \in \mu(i)} u_i(j)$. Similarly, for $j \in \mathcal{R}$, the value for matching μ is $v_j(\mu) = \sum_{i \in \mu(j)} v_j(i)$. Valuations

that are more general than additive valuations need to be specified for every feasible subset of partners. This can often make them hard to specify without access to an oracle. However, in this thesis we shall only consider those valuation functions whose values can be computed in time polynomial in the number of agents. The negative of valuations shall be called costs. We shall only consider costs when *all* agents have non-positive valuations.

2.1.2 One-sided Preferences

As discussed in the previous chapter, there are many settings where agents have to be matched to items. Here items can be captured by agents who are indifferent across all matching outcomes. We can now simplify much of the notation of two-sided matchings. We shall use $\mathcal{A}$ to denote the set of n agents and $\mathcal{G}$ to denote the set of m items. Here $\mathcal{A}$ corresponds to the set $\mathcal{R}$ and $\mathcal{G}$ to the set $\mathcal{L}$ which now represents items. Items can be thought of as agents that are indifferent between all matching outcomes. Items which give all agents non-negative values are called goods. The items for which all agents have non-positive values or costs are called chores or bads.

While most work on one-sided preferences has considered cardinal valuations, some work has also considered ordinal valuations. However, this thesis only considers cardinal valuations. To this end, we have a valuation function for each agent $i \in \mathcal{A}$, $v_i : 2^{\mathcal{G}} \to \mathbb{R}$. When multiple items are allowed to be matched to a single agent, such matchings are typically called allocations. Thus, an instance of an allocation setting is denote by the tuple $\langle \mathcal{A}, \mathcal{G}, (v_i)_{i \in \mathcal{G}} \rangle$.

Definition 2.1 (Allocation) *$A = (A_1, \cdots, A_n)$ is said if it is an n-partition of $\mathcal{G}$. That is, for any $i, j \in [n]$ s.t. $i \neq j$, $A_i \cap A_j = \emptyset$ and $\cup_{i \in [n]} A_i = \mathcal{G}$.*

We shall use Π^n to denote the space of all feasible allocations. The valuation of an agent $i \in \mathcal{A}$ for a set $S \subseteq \mathcal{G}$ is denote by $v_i(S)$. We shall use the term allocation to refer to a many-to-one matching with one sided preferences. A one-one matching with one-sided preferences will be called a matching only. The computation of matchings that maximize social welfare can be done in polynomial time whenever the valuations are additive. However it is non-trivial to maximize social welfare over the space of fair allocations . We now define some properties which we would like for the resulting outcome to satisfy.

2.2 Efficiency

Beginning with efficiency, we first define the most commonly used definition of efficiency. For the purposes of this section, we will use the notation of two-sided matchings, but it can easily be extended to one-sided matchings.

Definition 2.2 (Pareto Optimality (PO)) *A matching μ Pareto dominates if for all agents $i \in \mathcal{L}$ $u_i(\mu) \geq u_i(\mu')$ and for all agents $j \in \mathcal{R}$ $v_j(\mu) \geq v_j(\mu')$ and there exists some agent $a \in \mathcal{L} \cup \mathcal{R}$ for whom the value from μ is* strictly greater *than the value from μ'. An outcome is Pareto optimal if it is not Pareto dominated by any other outcome.*

This is a well studied property, however many outcomes can satisfy it. Thus, there has been much study on stronger notions of efficiency. Ideally, we would want an outcome which maximises the sum total of the agents' values. The sum of all agent values for a given matching is called its social welfare. The outcome that maximizes this is said to be *socially optimal*.

Definition 2.3 (Social Optimality (SO)) *A matching $(s_1, \cdots, s_n)$ is socially optimal if $SW(s_1, \cdots .s_n) = \sum_{i=1}^{n} u_i(s_1, \cdots, s_n)$ is the maximum over all possible outcomes.*

Clearly, any outcome that satisfies SO, must also satisfy PO. While Pareto optimality can easily be defined for settings with ordinal preferences only, social welfare cannot. Given that the majority of work on two-sided matchings considers ordinal preferences, the way efficiency was captured, was through stability.

2.2.1 Stable Matchings

Stable matchings were first introduced in the seminal work of Gale and Shapley [52]. Since then, there have been multiple definitions of stability in various contexts but for the purposes of this thesis, we shall define stability as the absence of blocking pairs. We extend the typical definition to matchings with cardinal valuations as follows:

Definition 2.4 (Blocking Pair) *An agent pair (i, j) with $i \in \mathcal{L}$ and $j \in \mathcal{R}$ form a blocking pair for μ if they both prefer each other more than one of their partners under μ. That is there exist $i' \in \mu(j)$ and $j' \in \mu(i)$ such that $v_j(i') < v_j(i)$ and $u_i(j') < u_i(j)$.*

Definition 2.5 (Stable Matching) *A matching μ is said to be stable if there does not exist any blocking pair under μ.*

It is straightforward to see that under strict preferences, stability does in fact imply Pareto optimality. For a one-one bipartite matching setting, Gale and Shapley [52] gave an efficient algorithm to find a stable matching. The algorithm essentially chooses one side as the proposing side and the other as the accepting side. An unmatched agent in the proposing side proposes to the agent on the other side whom she most prefers out of those she hasn't proposed to. An agent on the accepting side accepts a proposal if she is unmatched or she

prefers the proposing agent to the one she is matched with. This process repeats till there are no unmatched agents.

This translated to what is popularly known as the Deferred Acceptance mechanism to find a stable matching for a many-to-one matching setting. An important property of the Gale-Shapley algorithm is that it always computes an optimal matching for the proposing side. That is, for any agent on the proposing side, she is matched to her most preferred out of any partner under a stable matching. For agents on the accepting side, however, the converse happens, where each agent on the accepting side is matched to her least preferred partner out of all partners she is matched to under a stable matching.

Foundational Results on Stable Matchings

Roth [101] found that while no matching mechanism that computes a stable matching can be incentive compatible for all agents, the Gale Shapley mechanism is incentive compatible for the proposing side. This sparked a long series of investigations on manipulating the Gale Shapley mechanism [66, 67, 110, 115, 119]. An important result was by Vaish and Garg [119] where they looked at optimal manipulations for a single agent under the Gale Shapley mechanism. They showed that lattice of stable matchings when one agent reports her optimal manipulation and all others are truthful is a subset of the lattice of stable matchings under the original instance. Consequently, whenever a matching instance has a unique stable matching, the Gale Shapley mechanism is DSIC for all agents involved.

Another important set of results is on the structure of the space of stable matchings. Starting with the work of Vande Vate [121] who showed that the space of stable matchings has a linear programming formulation, followed by subsequent refinements of this result [103, 105, 114]. This important result not only gave intuition into the space of stable matchings, but also provided a tool to optimize over this space for various desirable objective functions. As a result of this work, we can efficiently optimize for for various desirable criteria like social welfare and other welfare functions like Nash welfare over this space.

While stability has been extensively studied for matchings under two sided preferences, under one-sided preferences, it does not make much sense. Every matching or allocation is in fact stable as items have no reason to form blocking pairs. Thus, for these settings notions like Pareto Optimality and Social Optimality make more sense. That said, stability continues to be an extremely desirable, and often non-negotiable, property for two-sided preferences.

2.3 Fair Allocations

Stability became so integrated in the framework of two-sided matchings, the attempts at studying fairness in this space was also restricted to the space of stable matchings. However, for one-sided matchings, fairness is equally crucial, increasingly so, in recent years. To this end, in the past decade, the computational problem of fairly dividing indivisible items among agents (essentially matchings with only agents on one side having preferences) has received enormous attention by the EconCS research community [17, 19, 30, 33, 36, 41, 96]. While the last decade has seen a surge in the work done in fair allocations, the study was initiated much earlier with fair cake cutting [112]. The most plausible fairness notion is often considered to be *envy-freeness*.

2.3.1 Envy-freeness and Extensions

Definition 2.6 (Envy-free) *Given a fair allocation instance* $\langle \mathcal{A}, \mathcal{G}, (v_i)_{i \in \mathcal{G}} \rangle$ *an allocation* $A = (A_1, \cdots, A_n)$ *is said to be envy-free if for any* $i, j \in \mathcal{A}$, $v_i(A_i) \geq v_i(A_j)$.

In the setting of indivisible items, (see Amanatidis et al. [5], Aziz et al. [9] for a recent survey of the area), envy-free solutions need not exist. The closest relaxation is envy-free upto any item (EFX).

Definition 2.7 (EFX) *Given a fair allocation instance* $\langle \mathcal{A}, \mathcal{G}, (v_i)_{i \in \mathcal{G}} \rangle$ *an allocation* $A = (A_1, \cdots, A_n)$ *is said to be* EFX *if for any* $i, j \in \mathcal{A}$, *for all* $g \in A_j$ *we have that* $v_i(A_i) \geq v_i(A_j \setminus g)$.

The existence of the EFX objective is still not resolved for the general case of any number of agents and items. A relaxation of envy-freeness that exists in standard settings is *envy-freeness upto one item* (EF1).

Definition 2.8 (EF1) *Given a fair allocation instance* $\langle \mathcal{A}, \mathcal{G}, (v_i)_{i \in \mathcal{G}} \rangle$ *an allocation* $A = (A_1, \cdots, A_n)$ *is said to be* EF1 *if for any* $i, j \in \mathcal{A}$, *there exists* $g \in A_j$ *s.t.* $v_i(A_i) \geq v_i(A_j \setminus g)$.

It was defined by Budish [33] (and, implicitly, a few years earlier by Lipton et al. [82]). In contrast to envy-freeness which is usually impossible to achieve, EF1 is always achievable in the standard setting and is also compatible with notions of economic efficiency [15, 35]. These papers assume that items are goods, i.e., agents have non-negative valuations for them. An important result from this space is from Caragiannis et al. [35] showing that an EF1 and Pareto Optimal allocation is guaranteed to exist. Non-positive valuations, i.e., indivisible chores, have also received attention. More importantly, a series of recent papers consider mixed items that can be goods for some agents and chores for others [8, 22, 24].

Envy Graphs. A common approach to finding allocations that satisfy some relaxation of envy-freeness uses envy graphs [8, 14, 27, 82, 96]. The envy graph of an allocation A refers to a directed graph where there is a single vertex for each agent $i \in \mathcal{A}$ and there is an edge between i and j if i prefers the bundle allocated to j , A_j over their own bundle A_i. In this thesis, we find such approaches instrumental in both one-sided and two-sided matching settings.

Envy-graph based algorithms typically start with an empty bundle allocated to each agent and then pick either a source, if allocating a good, or a sink, if allocating a chore, of the envy graph to give the next item to. Any cycle on the envy graph can be "resolved" by giving each agent on the cycle the bundle currently given to their successor on the cycle. This is repeated till all items are allocated. This is typically called cycle elimination. Algorithms employing these techniques will usually maintain a property as an invariant and can sometimes require some additional processing either before or after cycle elimination, before allocating the next item.

2.3.2 Maximin and Leximin

While envy-based fairness certainly has its merits, it can often lead to highly inefficient solutions, as eliminating envy often requires some degree of inefficiency. Another approach to fairness is by using a threshold based notion, trying to optimize the guarantee for the worst off agent. For goods it is called the Maximin Share. It aims to ensure that each agent gets at least as much value as they would if they were to create a partition of the items then receive their least preferred of the bundle. We first define the value of the MMS threshold.

Definition 2.9 (Maximin Share) *We say that ν_i is the maximin share (threshold or value) of $i \in \mathcal{A}$, if $\nu_i = \max_{A \in \Pi^n} \min_{j \in [n]} v_i(A_j)$.*

Definition 2.10 (MMS) *We say that an allocation $A = (A_1, \cdots, A_n)$ satisfies* MMS *if for all $i \in [n]$, $v_i(A_i) \geq \nu_i$ which is the maximin share threshold.*

MMS has been investigated thoroughly for the case of goods [10, 17, 57, 63, 98]. It is known that while an allocation satisfying MMS need not exist, an α-MMS is guaranteed to exist, for $\alpha = 3/4 + \frac{1}{12n}$ [55].

Definition 2.11 (α-MMS) *We say that an allocation $A = (A_1, \cdots, A_n)$ satisfies $alpha$-*MMS *if for all $i \in [n]$, $v_i(A_i) \geq \alpha\nu_i$ for some fixed $alpha \in [0, 1)$.*

A stronger notion of fairness is leximin optimality. The most convenient way of identifying such a matching is using leximin tuples. The leximin tuple of any allocation or matching is simply the tuple containing the values of all the agents under this matching, listed in non-decreasing order. Hence, the position of an agent's valuation in the leximin tuple may change under different matchings. The leximin tuple of a solution A will be denoted by $\mathcal{L}_A$. The t^{th} index of $\mathcal{L}_A$ is denoted by $\mathcal{L}_A[t]$.

Definition 2.12 (Leximin Domination) *We say that A leximin dominates A' if there exists a valid index k such that $\mathcal{L}_A[k'] = \mathcal{L}_{A'}[k']$ for all $k' < k$ and $\mathcal{L}_A[k] < \mathcal{L}_{A'}[k]$.*

We shall say that the leximin value of A is greater than that of A' if A leximin dominates A'. This shall be denoted by $\mathcal{L}_A > \mathcal{L}_{A'}$. We shall say that A is leximin inferior to A', when A' leximin dominates A.

Definition 2.13 (Leximin Optimal) *A leximin optimal solution A^* is one that is not leximin dominated by another allocation. In other words, A^* first maximizes the value of the worst-off agent, then maximizes the value of the second worst-off agent and so on.*

It is easy to see that a leximin optimal allocation will always be pareto optimal. Further, when agents have identical valuations, with no item having a zero marginal value to any set, the leximin optimal is also guaranteed to be EFX. The hardness of finding the leximin optimal allocation was established in [23, 96]. To the best of our knowledge, approximations to leximin have not been defined beyond the special case of two agents, where the problem reduces to egalitarian welfare. Bezakova and Dani [23] establish the APX-Hardness, for any factor better than $1/2$, of maximizing egalitarian welfare, or the Santa Claus problem. Here we aim to maximize the minimum value of any agent. Clearly, the leximin optimal does this and hence the hardness result extends to leximin.

An important assumption for our algorithms to be able to bypass the hardness is rankings. Prior fair division literature has called such preferences as same order preferences [30], fully correlated valuation functions[4] or instances with such preferences as ordered instances [10, 55]. Such preferences also generalize other well studied subclasses of preferences like identical valuations [11, 16] and single parameter environments [17]. To the best of our knowledge such settings have not previously been studied for leximin.

In special case of dichotomous valuations [29, 77], leximin optimality can be achieved in polynomial time. Bogomolnaia and Moulin [29] show that the maximum weight matching is stable under dichotomous preferences and satisfies a variety of properties including leximin

optimality. Recent results by Benabbou et al. [21] and Chen and Liu [41] study the properties of a leximin optimal allocation for restricted settings. Leximin has been used to ensure fairness in sortition [45, 46] and shows significant improvement on the systems currently in use.

2.4 Fairness in Two-Sided Matchings

Fairness in two-sided matching settings has often been defined from context-specific angles, such as college admissions [81, 93, 123, 126], focusing only on the colleges and not on the students. Our work looks at fairness from a broader perspective. Some prior literature has focused on combating the inherent bias towards the proposing side in the Gale Shapley algorithm [32, 68, 73, 109]. There is also some work on procedural fairness of the matching algorithms [74, 118].

However, these notions of fairness can be seen as group fairness notions and consider the two groups of agents, rather than ensuring fairness for each agent individually. Further, matching literature has almost exclusively considered settings with ordinal preferences, and looks for ordinal fairness notions. We consider cardinal valuations and adopt *almost envy-freeness* and *leximin optimality* from fair allocation literature. It is important to note that envy is typically defined very differently in matching settings. "Justified" envy has been studied for the many-to-one matching setting, such as that of school choice, and is a relaxation of stability in these settings as in [6, 122, 123, 126]. This is not the same as the notion of envy-freeness commonly studied in fair division literature such as [33, 47, 82, 112, 120] among many others.

Some recent work has looked at envy-based fairness in many-to-many matchings [49, 59], in contrast to our many-to-one matching setting, and does not require stability. Both of these use EF1 as their notion of fairness, we find that this need not coexist with stability, as discussed in Appendix B.1. A very recent paper by Karni et al. [71] formulates a notion of stability which is natural for many-to-one matchings under randomized mechanisms and seeks to achieve fairness. However, Karni et al. [71] look at fairness for only those agents that are matched to multiple agents (colleges) whereas we look at fairness for all agents. Moreover, Karni et al. [71] take as input ordinal valuations and not cardinal, and further imposes certain restrictions on the preferences of the colleges; the notion of fairness also needs metrics on the set of students and the set of colleges. We note that in contrast to Karni et al. [71], we look at deterministic algorithms with cardinal valuations and seek to satisfy fairness for all the agents involved.

It is also important to note that the notion of justified envy-freeness studied in prior work [7, 122, 124] is different from the envy-freeness notion studied in allocation literature. Justified envy-free means no blocking pairs, and is also called pairwise stability or just, stability, as in this thesis.

Part 1: One-Sided Preferences

Chapter 3

Fair and Efficient Matching of Delivery Tasks to Agents

We initiate the study of fair distribution of delivery tasks among a set of agents wherein delivery jobs are placed along the vertices of a graph. Our goal is to fairly distribute delivery costs, captured by a submodular function, among a fixed set of agents while satisfying some desirable notions of economic efficiency. We adopt well-established fairness concepts, *envy-freeness up to one item* (EF1) and *minimax share* (MMS) to our setting. We show that fairness is often incompatible with the efficiency notion of *social optimality*. Then we characterize instances that admit fair and socially optimal solutions by exploiting graph structures. It turns out that achieving fairness in conjunction with Pareto optimality is computationally intractable. This is complemented by designing an XP algorithm (parameterized by the number of agents) for finding MMS and Pareto optimal solutions on every instance. The same algorithm can be modified to find efficient solutions along with EF1, when such solutions exist. We complement our theoretical results by experimentally analyzing the price of fairness on randomly generated graph structures.

3.1 Introduction

With the rise of digital marketplaces, package delivery services have become crucial components of a multitude of e-commerce platforms such as Amazon, Flipkart, and eBay. In addition to these novel platforms, traditional postal and courier services also require swift turnarounds for distributing packages. Prior work has extensively investigated the optimal partitioning of

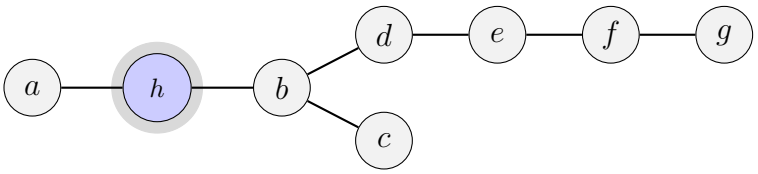


Figure 3.1: Example graph

tasks among the delivery agents under the guise of *vehicle routing* problems (see [116] for an overview). However, these solutions are primarily focused on optimizing the efficiency (often measured by delivery time or distance travelled [75, 95]), and do not consider fairness towards the delivery agents. This is particularly important in the settings where agents do not receive monetary compensation, e.g., in volunteer-based social programs such as Meals on Wheels [94] or where agents earn a fixed monthly income like postal workers.

In this chapter, we consider the fair distribution of delivery orders that are located on the vertices of a connected graph, containing a warehouse (the *hub*). Agents are tasked with picking up delivery packages (or *items*) from the hub, deliver them to the vertices, and return to the hub. In this setting, the cost incurred by an agent i is the total distance traveled, that is, the total number of distinct edges traversed by i in the graph. This makes the costs submodular. Let us illustrate this through an example.

Example 3.1 *Let there be seven delivery orders* $\{a, b, \ldots, g\}$ *and a hub* (h) *that are located on a graph as depicted in Figure 3.1. An agent's cost for servicing a set of orders or vertices depends on the graph structure and is* submodular. *For instance, the cost of delivering an order to vertex* f *is the distance from the hub* h *to* f*, which is* 4 [1]*; but the cost of delivering to* f *and* g *is only* 5 *since they can both be serviced in the same trip.*

*Let there be two (delivery) agents. If the objective were to simply minimize the total distance travelled (*social optimality*), then there are two solutions with the total cost of* 7*: either one agent delivers* all *the items or one agent services* a *while the other services the rest. However, these solutions do not distribute the delivery orders fairly among the agents.*

One fair solution may assign $\{a, b, f\}$ *to the first agent and* $\{c, d, e, g\}$ *to the other, minimizing the cost discrepancy. However, since both agents benefit from exchanging* f *for* c*, this allocation is not efficient or, more precisely, not* Pareto optimal. *After the exchange, the first agent services* $\{a, b, c\}$ *and the second agent* $\{d, e, f, g\}$*, which in fact is a Pareto optimal allocation.*

[1] Formally, there is also the cost of returning to h, but since, on trees, each edge must be traversed by an agent twice (once in each direction), we do not count the return cost for simplicity.

	–	PO	SO
EF1			
existence	✓	✗ (Prop. 3.6)	✗ (Thm. 3.2)
computation	P (Prop. 3.1)	NP-h (Prop. 3.4)	NP-h (Prop. 3.2)
MMS			
existence	✓	✓ (Prop. 3.3)	✗ (Thm. 3.3)
computation	NP-h (Thm. 3.1)	NP-h (Prop. 3.4)	NP-h (Prop. 3.2)

Table 3.1: Resolving the existence of fair and efficient delivery allocations

The above example captures the challenges in satisfying fairness in conjunction with efficiency, and consequently, motivates the study of fair distribution of delivery orders. The literature on fair division has studied many different fairness concepts, EF1 and MMS, among the more popular ones. A key question is how to adopt these fairness concepts to the delivery problems, and whether these fairness concepts are compatible with natural efficiency requirements.

3.1.1 Technical Contributions of this Chapter

In this chapter, we initiate the study of fair distribution of delivery tasks among a set of agents. The tasks are placed on the vertices of a graph and have submodular service costs. The primary objective is to find a fair partition of m delivery orders (represented by vertices of a tree), starting from a fixed hub, among n agents. We restrict our attention to tree graphs as they allow for tractable routing solutions (unlike cyclic graphs [70]). Despite this, these acyclic, undirected graphs provide a rich framework for studying fair and efficient solutions. We consider two well-established fairness concepts: EF1 and MMS. For both, we explore their existence and computation along with efficiency notions of social optimality (SO) and Pareto optimality (PO). Table 3.1 summarizes of our results. ✓ denotes that the allocation always exists, and ✗ that it need not exist. For every ✗ we provide a counterexample of such a combination of fairness and efficiency not existing.

We first show that an EF1 allocation always exists and can be computed in polynomial time (Proposition 3.1). In contrast, while an MMS allocation is guaranteed to exist, its computation remains NP-hard (3.1). While fair socially optimal solutions may not always exist, by exploiting the structure of the graph we provide conditions for which a socially optimal solution exists (Proposition 3.5), which implies a characterization for the existence of EF1 and SO allocations (Theorem 3.2) as well as MMS and SO allocations (Theorem 3.3).

Despite the intractability of satisfying MMS, we design an XP algorithm (Algorithm 3.2) that computes the Pareto frontier of a given instance in polynomial time when the number of agents is constant (Theorem 3.5). It allows us to find an MMS and PO solution and check whether an instance admits an EF1 and PO allocation (Theorem 3.6). This is a consequence of Theorem 3.4, in which we show that, intriguingly, while an EF1 and PO allocation does not always exist, when such an allocation exists it must satisfy MMS.

We complement our theoretical findings with detailed experiments on randomly generated graph structures. Specifically, we check how often an EF1 and PO allocation exists and we analyze the price of fairness, i.e., the increase in the total cost of agents that is needed to obtain a fair solution.

3.1.2 Relevant Prior Work

A primary distinction of this model from more standard fair division settings is that we consider submodular costs or supermodular values. On the other hand, submodular valuations and their subclasses have been well studied in prior work. Typically, submodular valuations require oracle access as the functions tend to be too large to be sent as an input to the algorithm. Depending on the type of oracle access, the abilities and efficiency of the algorithm changes. Submodular valuations and its subclasses have also been explored in fair division literature, especially in the context of MMS [10, 12, 13, 21, 57, 79]. Additionally, while the majority of the work in this space has looked at settings with goods, some recent work also looks at chores, either alone or in conjunction with goods [8, 24, 34, 48, 65, 68].

Submodular costs have been studied in various algorithmic settings, be it combinatorial auctions, facility location or other graph problems like shortest cycles. Of these the only study to look at a cost model similar to ours is [113] where the authors consider submodular facility costs using a rooted tree whose leaves are the facilities. The facility costs of opening a certain set of facilities was the sum of the weights of the vertices needed to be crossed from the root to reach reach these facilities from the root. While this is very similar to the cost functions in our model, it is important to note that this work has no fairness considerations, only aiming to minimise the total costs. Needless to say, this can cause a large discrepancy in the workloads of different facilities.

Fairness with Graphs

Some prior work has also looked at fair division on graphs [25, 31, 84, 85, 117]. The first difference of this body of work with our setting is that they solely consider items with non-negative values or goods. Secondly, with the exception of Misra and Nayak [84], all other

papers looked at settings where items are placed on the vertices of a graph and each agent must receive a connected bundle. Our model does not have such a restriction. The purpose behind the prior work done in fair division on graph is to partition the graph into n vertex-disjoint connected subgraphs. The value of the agents comes only from the vertices in their bundle, and does not depend on vertices outside of their bundle. The relaxations of envy are also based on removing items that lie on the boundary of the sets.

In our case, the subgraphs traversed by the agents will always intersect in the hub. Additionally, for us, the bundle that an agent receives need not form a connected subgraph. The cost depends on the distances of these vertices to the hub, which is in not allocated. Thus, agents can incur costs from having to traverse vertices that are not in their bundles. Misra and Nayak [84] look at a setting where agents are connected by a social network and only envy neighboring agents.

Work on fairness in delivery settings is scarce and almost entirely empirical [61, 87]. Further, no positive theoretical guarantees have been provided. The aim is typically to achieve fairness by income distribution, which differs from our approach. We look for fairness in the workload of the delivery agents, as there can be many settings where agents receive no or fixed monetary compensation for their work.

While fairness has been studied in routing problems, the aim has been to balance the amount of traffic on each edge [75, 95]. This does not capture the type of delivery instances that we look at. Some work has also looked at ride-hailing platforms and aims to achieve group and individual fairness in these settings [42, 106]. However, these studies are largely experimental and do not provide any theoretical guarantees. Further, these models also do not look at models with submodular costs. In fact, they use linear programs to achieve their experimental results.

There is a large body of work that looks at supply chain optimization [28, 54, 56, 88, 89]. The focus is to make each portion of the supply chain more efficient. In our setting, we assume that the equipment and logistic technicalities of the supply chain are already fixed. The study of fairness in this area focuses largely on fair profit distribution [40, 100, 107, 125]. We can now discuss the relevant notations and definitions needed to present our work.

3.2 Notation and Preliminaries

In our setting, n delivery agents must collectively deliver m delivery orders/packages. Each agent starts from the hub h to deliver any assigned order. Each of the order and the hub lie on a distinct vertex on the undirected acyclic graph or tree $\mathcal{G} = (X, E)$ which is rooted at h. Further, the vertex set X has cardinality $m + 1$, i.e. there is a vertext for each of the m orders

and one for the hub h. Thus, we denote a *delivery instance* by the ordered triple $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$. We present some basic graph definitions including paths, walks, etc. in Section 3.2.1. The goal is to assign each vertex in graph $\mathcal{G}$, except for the hub, to a unique agent that will *service* it. For this setting, an allocation $A = (A_1, \ldots, A_n)$ is an n-partition of vertices in $X \setminus \{h\}$. We are interested in *complete* allocations such that $\cup_{i \in \mathcal{A}} A_i = X \setminus \{h\}$.

An agent's *cost* for servicing a vertex $x \in X$, denoted by $c(x)$, is the length of the shortest path from the hub h to x. An agent's cost for servicing a set of vertices $S \subseteq X \setminus \{h\}$ is equal to the minimum length of a walk that starts and ends in h and contains all vertices in S divided by two. On trees, in each such walk, each edge is traversed by an agent twice, once in both directions. For simplicity, we drop the return cost, hence the division by 2. A walk that visits the vertices in S may form a superset of S, i.e. $S' \supseteq S$. Thus, the cost function c is monotone and *submodular* and belongs to the class of submodular *coverage functions*.

We use $G|_S$ to denote the minimal connected subgraph containing all vertices in $S \cup \{h\}$. Thus, we have $c(S) = |E(G|_s)|$. We focus our attention on trees. For arbitrary cyclic graphs, computing the minimum length of the walk is NP-hard [70]. We say that an agent servicing S, *visits* all vertices in $G|_S$. We first give the necessary graph definitions.

3.2.1 Graph Preliminaries

We now give relevant graph preliminaries for the work done in this chapter.

Definition 3.1 (Graph) *A* graph *is defined by a pair $\mathcal{G} = (X, E)$, where V (or $X(\mathcal{G})$) is a set of* vertices *and E (or $E(\mathcal{G})$) is a set of (undirected)* edges.

Definition 3.2 (Walk) *A* walk *is a sequence of vertices $(x_0, \ldots, x_k)$ such that every two consecutive vertices are connected by an edge.*

The *length* of a walk is the number of edges in it, i.e. the number of (distinct) vertices minus 1.

A *path* is a walk in which all vertices are pairwise distinct.

Definition 3.3 (Connected Graph) *A graph is* connected *if there exists a path between every pair of vertices in X.*

In a connected graph, the distance between vertices u and v, i.e., $\mathrm{dist}(u, v)$, is the minimum length of a path connecting u and v.

Definition 3.4 (Subgraph) *A* subgraph *of graph $\mathcal{G}$ is any graph $\mathcal{H} = (U, M)$ such that $U \subseteq X$ and $M \subseteq E$.*

A subgraph is *induced* (by a set of vertices U) if for every edge $(u, v) \in E$ such that $u, v \in U$ we have $(u, v) \in M$.

Definition 3.5 (Tree) *A* tree *is a graph in which between every pair of vertices there exists exactly one path.*

A tree can alternately be recognized as a connected graph with the number of edges being one less than the number of vertices or a connected acyclic graph. A *rooted tree* is a tree with one vertex, $h \in X$, designated as a *root*. For every vertex $u \in V$, its *ancestor* is every vertex in a path from u to h except for u. For such vertices u is a *descendant*. An ancestor (or descendant) of a vertex connected to it by an edge is called a *parent* (or *child* respectively). A vertex without children is called a *leaf*. A vertex that is not a leaf is called an *internal vertex*. A *subtree rooted in* u is an induced subgraph $T_u = (U, M)$ rooted in u, where U contains u and all its descendants. By a *branch* outgoing from h we understand a set of nodes of a subtree rooted in a child of the root h.

We are now equipped to define fairness and efficiency in this model.

3.2.2 Fairness Concepts

Here we shall show how we adapt popular fairness notions to our setting. We adapt *envy-freeness up to one order* (EF1) [33, 82] as for every pair of agents, if one agent envies another agent, then the envy can be eliminated by the removal of one vertex (delivery order) from the allocation of the envious agent.

Definition 3.6 (Envy-Freeness up to One Order (EF1)) *An allocation* $A = (A_1, \cdots, A_n)$ *is* EF1 *if for every pair* $i, j \in \mathcal{A}$*, either* $A_i = \emptyset$ *or there exists* $x \in A_i$ *such that* $c(A_i \setminus \{x\}) \leq c(A_j)$.

From prior investigations [15, 17, 35], EF1 has proven to be tractably computable and quite compatible with efficiency notions like SO and PO. In our model, we find that some results for EF1 in standard fair division settings extend but several do not. In comparison to EF1, MMS is harder to satisfy, even independently, in typical fair allocation settings.

In this setting, where none of the orders give positive values, MMS can be relabelled as *minimax share* (MMS), which ensures that each agent gets at most as much cost as they would if they were to create an n-partition of the delivery orders but then receive their least preferred bundle.

Definition 3.7 (Minimax Share (MMS)) *An agent's minimax share cost is* $\text{MMS}_i(I) = \min_{A \in \Pi^n} \max_{i \in \mathcal{A}} c(A_i)$*, where* Π^n *is the set of all complete allocations. Given an instance* I*, allocation* $A = (A_1, \cdots, A_n)$ *is* MMS *if* $c(A_i) \leq \text{MMS}_i(I)$ *for every* $i \in \mathcal{A}$.

Under non-negative identical valuations, MMS implies EF1. We shall give a proof of this shortly. Recall that we consider positive costs, that is non-positive valuations. For this setting, we show in Example 3.2, that EF1 and MMS do not imply one another under submodular costs functions even when the cost functions are identical. We now give a summary of work on MMS and EF1 and how they relate to each other.

MMS and Envy-Freeness

The notion of the maximin fair share was introduced by [33], which was also the first time the notion of EF1 was explicitly defined ([82] define it implicitly). When valuations are identical, MMS allocations are guaranteed to exist, as the allocation that defines the MMS value will satisfy MMS for all agents. Unfortunately, MMS allocations need not exist when valuations are not identical. However, for additive goods, a $3/4$-MMS is guaranteed to exist [57]. For additive chores, a $4/3$-MMS allocation is guaranteed to exist [10].

It is well-known that EF implies MMS for additive items, via Proportional share. However, EF1 does not imply MMS, even for identical additive valuations. In fact even the stronger notion of *envy-freeness up to any item* (EFX) does not imply MMS for identical valuations. Take a simple example. Let there be two agents and four goods, g_1 and g_2 both giving a value of 2, and g_3 and g_4 giving a value of 1 each. Here an MMS allocation would give both agents a value of 3. Now, the allocation $(\{g_1, g_2\}, \{g_3, g_4\})$ is EFX (and hence, EF1) but clearly not MMS.

From Plaut and Roughgarden [96] we know that when valuations are identical and the marginal values are positive for all goods, the leximin optimal allocation will be EFX and hence EF1. Now the leximin optimal under identical values is always MMS, hence an allocation that is both MMS and EFX always exists here. Further, Barman and Krishnamurthy [10] give an algorithm that satisfies both EFX and $2/3$-MMS for identical additive valuations.

Unfortunately, even EF need not imply MMS under our setting. Consider the instance and allocation depicted in Figure 3.2. The visible allocation, i.e., $(\{a, c\}, \{b, d\})$, is EF but not MMS. Both agents incur a cost of 3, but the MMS cost is 2. In this model, we find an interesting relationship between efficient and EF1 allocations and MMS.

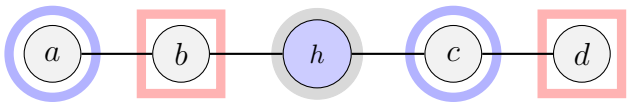


Figure 3.2: EF does not imply MMS

3.2.3 Economic Efficiency

We now show how we adapt standard efficiency notions to our setting. Our first notion of efficiency is social optimality that requires that the summed total cost of the agents is minimum. In this setting, it means that the summed total cost is equal to the number of edges in the graph. In such a case, each vertex (except for the hub) is visited by only one agent.

Definition 3.8 (Social Optimality (SO)) *An allocation $A = (A_1, \cdots, A_n)$ is socially optimal if $\sum_{i=1}^{n} c(A_i) = |E(G)|$. In other words, for every pair of agents $i \neq j \in \mathcal{A}$, the only vertex they both visit is the hub, i.e., $V(G|_{A_i}) \cap V(G|_{A_j}) = \{h\}$.*

An allocation that assigns all vertices to a single agent is trivially SO. Further, it is straightforward to verify if a given allocation is SO. However, as we discussed in Example 3.1 it may result in a rather unfair distribution of orders. Therefore, we consider a weaker efficiency notion that allows for some overlap in vertices visited by the agents.

Definition 3.9 (Pareto Optimality (PO)) *An allocation $A = (A_1, \cdots, A_n)$ Pareto dominates A' if for all agents $i \in \mathcal{A}$ $c(A_i) \leq c(A'_i)$ and there exists some agent $j \in \mathcal{A}$ such that $c(A_j) < c(A'_j)$. An allocation is Pareto optimal if it is not Pareto dominated by any other allocation.*

Observe that SO implies PO. Thus, it is simple to find *one* PO allocation, namely, one in which a single agent services all the orders. However, given an arbitrary allocation, checking if it is PO is not straightforward. We now follow up on Example 3.1 and analyze allocations satisfying our notions.

Example 3.2 *Consider an instance with 2 agents and the graph from Figure 3.1. As previously noted, there are only two* SO *allocations with minimum total cost, and neither is* EF1 *or* MMS. PO *allocation $(\{d, e, f, g\}, \{a, b, c\})$ is* MMS *(g must be serviced, hence, the* MMS *cost cannot be smaller than 5), but it is not* EF1. *In fact, there is no* EF1 *and* PO *allocation in this instance, as an agent servicing g has to service f, e and d as well (otherwise giving them to this agent would be a Pareto improvement). But then, even when we assign the remaining vertices, a, b, c, to the second agent, it will not be* EF1. *Finally, observe that allocation $(\{a, b, f\}, \{c, d, e, g\})$ is* EF1*, but not* MMS.

In order to study the (co-)existence of these fairness and efficiency notions, we use leximin optimality. In this setting, a leximin optimal allocation first minimizes the cost of the worst-off agent, then minimizes the cost of the second worst-off agent, and so on. We note that a leximin optimal allocation is always Pareto optimal.

3.2.4 Overview of Main Results

We can now give a formal summary of the results presented in this chapter. Firstly, we find that while both EF1 and MMS allocations exist independently, only finding EF1 allocations is computationally tractable.

Proposition 3.1 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* EF1 *allocation always exists and can be computed in polynomial time.*

Theorem 3.1 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* MMS *allocation always exists, however, finding such an allocation is* NP-hard.

From Example 3.2 we can see that allocations that are EF1 or MMS and SO need not exist. We further find that verifying whether they do for a given instance is NP-Hard.

Proposition 3.2 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* SO *allocation that satisfies* EF1 *or* MMS *need not exist. Moreover, checking whether an instance admits such an allocation is* NP-hard.

Despite this intractability, we provide a characterization for when fair and SO allocations exist.

Theorem 3.2 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, there exists an* EF1 *and* SO *allocation if and only if there exists a partition $(P_1, \ldots, P_n)$ of branches outgoing from h such that $\sum_{B \in P_i} |B| - \sum_{B \in P_j} |B| \leq 1$, for every $i, j \in \mathcal{A}$.*

Theorem 3.3 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, there exists an* MMS *and* SO *allocation if and only if there exists a partition $(P_1, \ldots, P_n)$ of branches outgoing from h such that $\sum_{B \in P_i} |B| \leq MMS_i(I)$ for every $i \in \mathcal{A}$.*

Moving on to Pareto Optimality, while EF1 and PO allocations need not exist from Example 3.2, MMS and PO allocations always do. We find that finding either combination is also intractable.

Proposition 3.3 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* MMS *and* PO *allocation always exists.*

Proposition 3.4 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, checking whether there exists an* EF1 *and* PO *or finding an* MMS *and* PO *allocation is* NP-hard.

We then give an interesting characterization for the existence of EF1 and PO allocations and prove an interesting relation between EF1 and MMS through it.

Theorem 3.4 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, every* EF1 *and* PO *allocation satisfies* MMS.

In our final technical result, we give an algorithm that can help check for the existence of fair and efficient allocations. This algorithm is in polynomial time when the number of agents is constant.

Theorem 3.5 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, Algorithm 3.2 computes its Pareto frontier and runs in time $O((n+2)! m^{3n+2})$.*

Theorem 3.6 *There exists an XP algorithm parameterized by n, that given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, computes an* MMS *and* PO *allocation, and can decide whether the instance admits* MMS *and* SO*,* EF1 *and* PO*, and* EF1 *and* SO *allocations.*

We complement these theoretical findings by detailed experimentation on the existence of fair and efficient solutions, the cost that fairness imposes on efficiency and the execution and running time of the algorithm.

3.3 Fair Allocations

In this section, we consider EF1 and MMS allocations without any efficiency requirement.

Proposition 3.1 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* EF1 *allocation always exists and can be computed in polynomial time.*

This is computed by Algorithm 3.1. Recall that our cost functions are submodular and *monotone*. Therefore, an EF1 allocation can be obtained by adapting an *envy graph* algorithm from [82]. It starts with each agent having an empty set. We then pick an agent who currently has minimum cost (i.e., a sink of the envy graph) and give them a vertex that will give minimum marginal cost out of the unassigned vertices. This is repeated till all vertices in $X \setminus \{h\}$ are allocated. The algorithm described above always returns an EF1 solution for chores as long as valuations are monotone. The proof is analogous to the proof in [82].

Algorithm 3.1: Computing an EF1 allocation

Input: Fair Delivery Instance $I = \langle \mathcal{A}, G, h \rangle$ with $|V| = m + 1$
Output: An allocation A

Initialize allocation A with $A_i \leftarrow \emptyset$;
$P \leftarrow V \setminus h$;
Let G_A be the envy graph of A;
while $P \neq \emptyset$ **do**
  $i \leftarrow sink(G_A)$;
  $x_i \leftarrow \text{argmin}_{x \in P}\, c(A_i \cup x) - c(A_i)$;
  $A_i \leftarrow A_i \cup x_i$ ▷ Give sink of envy graph a new task;

An MMS allocation in our setting always exists. It follows from the fact that agents have identical cost functions (an allocation that minimizes the maximum cost will satisfy MMS). However, finding such an allocation is NP-hard. To establish this, we first show the hardness of finding the MMS cost.

Theorem 3.1 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* MMS *allocation always exists, however, finding such an allocation is* NP-hard.

Proof: We give a reduction from 3-PARTITION. In this problem, we are given $3k$ positive integers $S = \{s_1, \ldots, s_{3k}\}$ that sum up to kT for some $T \in \mathbb{N}$. The task is to decide if there is a partition of S into k pairwise disjoint subsets, $P = P_1, \ldots, P_k \subseteq S$ such that the elements in each subset sum up to T. This problem is known to be NP-hard [70], even when the values of the integers are polynomial in k.

For each instance of 3-PARTITION let us construct a delivery instance with k agents. To this end, for each integer $s_i \in S$, let us take s_i vertices $x_i^1, \ldots, x_i^{s_i}$, which, with the hub, h, gives as a total of $3kT + 1$ vertices. Next, for every $i \in [3k]$, let us connect all consecutive vertices in sequence $h, x_i^1, \ldots, x_i^{s_i}$ with an edge to form a path. In this way, we obtain a graph that consists of the hub and $3k$ paths of different lengths outgoing from the hub (such graphs are known as *spider* or *starlike* graphs). See Figure 3.3 for an illustration. In this figure, for two agents, the MMS cost is equal to 12, as there is no partition of integers $3, 3, 3, 6, 6, 1$ into two subsets such that the integers in each sum up to 11.

Let us show that the minimax share cost in this instance is T, if and only if, there exists a desired partition in the original 3-PARTITION instance. If there is a partition P, consider allocation A obtained by assigning to every agent $j \in \mathcal{A}$, all vertices corresponding to integers in P_j, i.e., $A_j = \cup_{s_i \in P_j} \{x_i^1, \ldots, x_i^{s_i}\}$. In this allocation, the cost of every agent will be equal to T. Further, the maximum cost of an agent in *any allocation* cannot be smaller than T. If it

were, it would make the total cost smaller than the number of edges in the graph, which is not possible. Hence, the MMS cost is T.

Conversely, if we know that the MMS cost is T, we can show that a 3-Partition exists. Take an arbitrary MMS allocation A. Consider the leaves in the bundle of an arbitrary agent $j \in \mathcal{A}$, i.e., vertices of the form $x_i^{s_i} \in A_j$ for some $i \in [3k]$. Observe that to service each such leaf, agent j has to traverse s_i edges and these costs are summed when the agent services multiple leaves. Now, we know that the total cost of j is at most T, i.e., $c(A_j) \leq T$. Thus, the sum of integers corresponding to the leaves serviced by each agents is at most T. Hence, the leaves serviced by agents give us a desired partition P. □

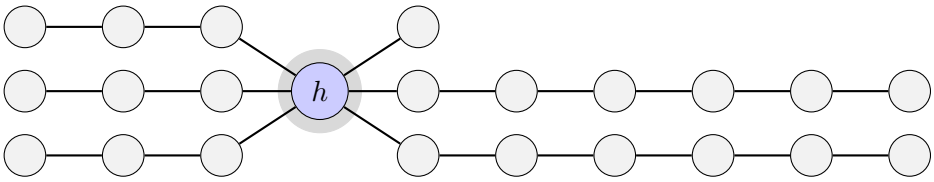


Figure 3.3: Example of a spider (or starlike) graph.

We have discussed the interactions of MMS with EF and its relaxations in various settings and demonstrate that in our setting EF need not imply MMS.

3.4 Characterizing Efficient and Fair Solutions

We have already established in Example 3.2 the possible incompatibility of fairness and efficiency in our setting. In this section, we characterize the space of delivery instances for which fair and efficient allocations exist. We first discuss social optimality and then turn our attention to Pareto optimality.

3.4.1 Social Optimality

We start with a simple characterization of all SO allocations, irrespective of their fairness.

Proposition 3.5 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an allocation A is SO, if and only if, every branch, B, outgoing from h, is fully contained in a bundle of some agent $i \in \mathcal{A}$, i.e., $B \subseteq A_i$.*

Proof: Given a fair delivery instance, for any allocation, observe that the sum of the costs of all agents can never be smaller than the number of edges in the graph, i.e., $|E| = m$. Indeed, since every vertex has to be serviced by some agent, each edge must appear in $G|_{A_i}$ for some $i \in \mathcal{A}$. Moreover, if every branch outgoing from the hub is fully contained in some bundle,

every edge appears in $G|_{A_i}$ for exactly one $i \in \mathcal{A}$. Hence, in every allocation satisfying the assumption, the total costs is equal to m and thus the allocation is SO.

Now, consider an allocation in which there exists a branch, B, and agents $i \neq j$ such that $B \cap A_i \neq \emptyset$ and $B \cap A_j \neq \emptyset$. Let u be a vertex in B connected to the hub, h. Then, edge (h, u) appears in both $G|_{A_i}$ and $G|_{A_j}$. Since every other edge appears in $G|_{A_k}$ for at least one $k \in \mathcal{A}$, we get that the total cost is greater than m. Thus, such an allocation is not SO. □

Proposition 3.2 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* SO *allocation that satisfies* EF1 *or* MMS *need not exist. Moreover, checking whether an instance admits such an allocation is* NP-hard.

Proof: Example 3.2 shows that there can exist instances where no SO allocation satisfies MMS or EF1. Now recall the proof of Theorem 3.1. Observe that the MMS allocations in the case when the original instance does admit a 3-Partition must also be SO. Hence, finding an MMS and SO allocation when one exists is NP-hard.

Now from Proposition 3.7 for any EF1 and SO allocation, $A = (A_1, \ldots, A_n)$, for any i, j, it must be that $|c(A_i) - c(A_j)| \leq 1$. As $\sum_{i=1}^{n} c(A_i) = nT$, there exist i s.t. $c(A_i) < T$ if and only if there exists i' such that $c(A_i) > T$. Thus, for such an i, EF1 is violated. Consequently, an EF1 and SO allocation must give cost T to all agents.

Hence, an EF1 and SO allocation exists if any only if a there exists a 3-Partition. □

Despite this, we exploit the graph structure of our setting to characterize EF1 and SO as well as MMS and SO allocations. To this end, we first characterize SO allocations in general in the following proposition.

Theorem 3.2 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, there exists an* EF1 *and* SO *allocation if and only if there exists a partition $(P_1, \ldots, P_n)$ of branches outgoing from h such that $\sum_{B \in P_i} |B| - \sum_{B \in P_j} |B| \leq 1$, for every $i, j \in \mathcal{A}$.*

Proof: From Proposition 3.5, we know that each SO allocation must be a partition of whole branches outgoing from h. Observe that if an agent's bundle consists of a union of whole branches, then removing a vertex from its bundle reduces the cost of the agent by 1, if the vertex is a leaf, or by 0, otherwise. Hence, in order to achieve EF1 the costs of agents cannot differ by more than one. □

We note that for most vertices in a tree the condition of Theorem 3.2 is not satisfied. In fact, the set of vertices for which this condition may hold can be characterized using a notion of the *center* of a graph.

Definition 3.10 *The* center *of a tree $\mathcal{G} = (X, E)$ is the maximal subset of vertices $C \subseteq X$ such that for every $x \in C$ it holds that*

$$\sum_{y \in X} \mathrm{dist}(x, y) = \min_{x' \in X} \sum_{y \in X} \mathrm{dist}(x', y),$$

where $\mathrm{dist}(x, y)$ is the length of a shortest path from x to y.

The center of the tree has been studied in computational social choice literature [111] and in theoretical computer science in general [58]. In particular, it is known that the center of a tree is always a single vertex or a pair of vertices connected by an edge. Moreover, if for some vertex v, there exists a partition of the branches outgoing from v such that the differences between the number of vertices in each bundle are equal or less than 1, then v has to be in the center. Thus, we obtain a following necessary (but not sufficient) condition.

Corollary 3.1 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, there exists an* EF1 *and* SO *allocation only if the hub is in the center of the tree.*

While deciding whether an EF1 and SO allocation exists is NP-hard (Proposition 3.2), the condition in Corollary 3.1 can be verified using a linear-time algorithm for checking if a vertex is in the center of a tree. Hence, we can say for example, that in the graph from Figure 3.1, an SO and EF1 allocation would be possible only if the hub was at vertex b or d.

Finally, as a consequence of Proposition 3.5, we obtain a characterization of MMS and SO allocations.

Theorem 3.3 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, there exists an* MMS *and* SO *allocation if and only if there exists a partition $(P_1, \ldots, P_n)$ of branches outgoing from h such that $\sum_{B \in P_i} |B| \leq MMS_i(I)$ for every $i \in \mathcal{A}$.*

Proof: From Proposition 3.5 we know that the allocation has to be a partition of whole branches outgoing from h. Hence, the theorem follows from the definition of MMS. □

3.4.2 Pareto Optimality

We now look at a slightly weaker notion of efficiency, Pareto optimality. Recall that all SO allocations are PO. The central result of this section is the proof that every EF1 and PO allocation will satisfy MMS as well. We first discuss the existence of PO allocations that also satisfy EF1 or MMS.

Proposition 3.6 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* EF1 *and* PO *allocation need not exist.*

This was demonstrated in Example 3.2. In contrast, MMS and PO solutions always exist.

Proposition 3.3 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* MMS *and* PO *allocation always exists.*

Proof: By definition, the leximin optimal allocation always exists. Since the cost functions are identical for all agents, this allocation will be MMS and PO by definition. □

We shall show that EF1 and PO allocations must also satisfy MMS. To this end, we first prove an insightful necessary condition for EF1 and PO allocations: in every EF1 and PO allocation, the difference in the cost of any pair of agents cannot be greater than 1.

Proposition 3.7 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$ and an* EF1 *allocation A, if $|c(A_i) - c(A_j)| > 1$ for some agents $i, j \in \mathcal{A}$, then A is not* PO.

Proof: Without loss of generality, assume that A is sorted in non-increasing cost order, i.e., $c(A_1) \geq \cdots \geq c(A_n)$ otherwise we can relabel the agents). We will show that if $c(A_n) < c(A_1) - 1$, then A is not PO, i.e., it is Pareto dominated by some allocation A' (not necessarily EF1).

For every vertex $x \in V \setminus \{h\}$, by $p(x)$ let us denote the *parent* of x in a tree G rooted in h. Also, for every agent $i \in \mathcal{A}$, let $w(i)$ be the *worst* vertex in i's bundle, i.e., the vertex which on removal gives the largest decrease in cost (if there is more than one we take an arbitrary one). Formally, $w(i) = \operatorname{argmax}_{x \in A_i} c(A_i) - c(A_i \setminus \{x\})$. Since A is EF1, for every agent i with maximal cost, i.e., such that $c(A_i) = c(A_1)$, we have

$$c(A_i \setminus \{w(i)\}) \leq c(A_n) < c(A_i) - 1. \tag{3.1}$$

Observe that this is only possible if the parent of $w(i)$ is not serviced by i, i.e., $p(w(i)) \notin A_i$.

In order to construct allocation A' which pareto dominates A, we look at the agent servicing the parent of the worst vertex of agent 1, call this agent i_1 . If i_1 incurs maximum cost, we look at the agent servicing the parent of the worst vertex of i_1. We continue in this manner and obtain a maximal sequence of pairwise disjoint agents $1 = i_0, i_1, \ldots, i_k$ such that $p(w(i_{s-1})) \in A_{i_s}$ and $c(A_{i_s}) = c(A_1)$ for every $s \in [k]$. The cost incurred by the agent servicing the parent of the worst vertex of i_k, which we denote by i^* (i.e., $p(w(i_k)) \in A_{i^*}$) can create two cases. Either i^* does not incur maximum cost, i.e., $c(A_{i^*}) < c(A_1)$ (Case 1), or it does and

it appears in the sequence, i.e., $i^* = i_j$ for some $j < k$ (Case 2).

Case 1. Consider allocation A' obtained from A by exchanging the bundles of agent i_k and agent i^* with the exception of $w(i_k)$ (which continues to be serviced by i_k). Formally,

$$A'_t = \begin{cases} A_t & \text{if } t \notin \{i_k, i^*\} \\ A_{i^*} \cup \{w(i_k)\} & \text{if } t = i_k \\ A_{i_k} \setminus \{w(i_k)\} & \text{if } t = i^* \end{cases}$$

Since costs of agents in $\mathcal{A} \setminus \{i^*, i_k\}$ are not affected, decreases without increasing the other's cost. To this end, observe that since parent of $w(i_k)$ belongs to A_{i^*}, adding this vertex to A_{i^*} increases the cost by 1, i.e.,

$$c(A'_{i_k}) = c(A_{i^*}) + 1. \tag{3.2}$$

Now, let us consider two subcases based on the original difference in costs of agents i_k and i^*.

Case 1a: If $c(A_{i_k}) > c(A_{i^*})+1$, then from Equation 3.2 we get that $c(A'_{i_k}) = c(A_{i^*})+1 < c(A_{i_k})$. Hence, the cost of i_k decreases. On the other hand, Equation 3.1 yields

$$c(A'_{i^*}) = c(A_{i_k} \setminus \{w(i_k)\}) \leq c(A_n) \leq c(A'_{i^*})$$

, so agent i^* does not suffer from the exchange.
Case 1b: $c(A_{i_k}) = c(A_{i^*}) + 1$. Now, the cost of i_k stays the same by Equation 3.2. However, since $c(A_n) < c(A_1) - 1$, it must be that $c(A_n) < c(A_{i^*})$. Thus, from Equation 3.1 we have

$$c(A'_{i^*}) \leq c(A_n) < c(A_{i^*}),$$

i.e., the cost of i^* decreases.

Case 2. When $i^* = i_j$ for some $j < k$, we have a cycle of agents $i^* = i_j, i_{j+1}, \ldots, i_k$ such that $c(A_{i_s}) = c(A_1)$ and $p(w(i_s)) \in A_{i_{s+1}}$ for every $s \in \{j, \ldots, k\}$ (we denote i_j as i_{k+1} as well for notational convenience). Here, we consider two subcases, based on whether somewhere in the cycle the parent of the worst vertex of one agent is the worst vertex of the next agent, i.e., $p(w(i_s)) = w(i_{s+1})$ for some $s \in \{j, \ldots, k\}$.

Case 2a: there exists s s.t. $p(w(i_s)) = w(i_{s+1})$ for some $s \in \{j, \ldots, k\}$.
Consider A' obtained from A by giving $w(i_{s+1})$ to agent i_s. Formally,

$$A'_t = \begin{cases} A_t & \text{if } t \notin \{i_s, i_{s+1}\} \\ A_{i_{s+1}} \setminus \{w(i_{s+1})\} & \text{if } t = i_{s+1} \\ A_{i_s} \cup \{w(i_{s+1})\} & \text{if } t = i_s \end{cases}$$

Now, the cost of agent i_{s+1} decreases as it no longer services its worst vertex. Since agent i_s was servicing a child of $w(i_{s+1})$, it was visiting $w(i_{s+1})$ on the way. Hence, the cost of i_s stays the same. As the bundles of the remaining agents did not change, A' Pareto dominates A.

Case 2b: for no s, $p(w(i_s)) = w(i_{s+1})$ for some $s \in \{j, \ldots, k\}$.
Finally, if in the cycle there is no agent for which the parent of its worst vertex is the worst vertex of the next agent, we swap the worst vertices along the cycle. Formally,

$$A'_t = \begin{cases} A_t & \text{if } t \notin \{j+1, \ldots, k+1\} \\ A_{i_t} \setminus \{w(i_t)\} \cup \{w(i_{t-1})\} & \text{if } t \in \{j+1, \ldots, k+1\} \end{cases}$$

Since each agent i_s is servicing the parent of the worst vertex of the previous agent in the cycle, i.e., $p(w(i_{s-1})) \in A'_{i_s}$, servicing $w(i_{s-1})$ incurs an additional cost of 1. However, from Equation 3.1 giving away the worst vertex decreases the cost of more than 1. Hence, the cost of each agent in the cycle decreases. The other agents' costs stay the same, thus A' Pareto dominates A.

□

We now show that EF1 and PO imply MMS.

Theorem 3.4 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, every* EF1 *and* PO *allocation satisfies* MMS.

Proof: We will actually show a stronger result: that an EF1 and PO allocation A is necessarily leximin optimal (thus, MMS as well). Without loss of generality, assume that A is sorted in non-increasing cost order, i.e., $c(A_1) \geq \cdots \geq c(A_n)$ (otherwise we can relabel the agents). By contradiction, suppose that there exists A' (also sorted) that leximin dominates A, i.e., there exists $i \in \mathcal{A}$ such that $c(A'_i) < c(A_i)$ and $c(A'_j) = c(A_j)$ for every $j \in [i-1]$. Fix an agent $j \in \mathcal{A}$ such that $j > i$. From Proposition 3.7 we know that its cost is either $c(A_j) = c(A_i)$

or $c(A_j) = c(A_i) - 1$. On the other hand, we assumed that $c(A'_j) \leq c(A'_i) < c(A_i)$, hence $c(A'_j) \leq c(A_j)$. Thus, A' Pareto dominates A, which is a contradiction. □

On the other hand, from Example 3.2, MMS, even with PO, does not imply EF1. From the proof of the above theorem we obtain the following remark.

Remark 1 *Given a fair delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* EF1 *and* PO *allocation is leximin optimal.*

Combining this with proposition 3.7 we obtain the following characterization.

Corollary 3.2 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, an* EF1 *and* PO *allocation exists if and only if for every leximin optimal allocation A, $\max_{i,j \in \mathcal{A}} |c(A_i) - c(A_j)| \leq 1$.*

In Section 3.6, we experimentally analyze how often EF1 and PO solutions do exist on randomly generated graphs.Finally, by modifying the proof of Theorem 3.1 we show that finding a PO allocation that is MMS or EF1 (and deciding if PO and EF1 allocation exists) is NP-hard.

Proposition 3.4 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, checking whether there exists an* EF1 *and* PO *or finding an* MMS *and* PO *allocation is* NP-hard.

Proof: Recall the proof of Theorem 3.1 and Proposition 3.2. In the instances constructed, we shall now show that any PO allocation must be SO.

Let A be an allocation that is not SO. Thus, there exists vertex x_j^t that is visited by multiple agents. Clearly, x_j^t is not a leaf. Recall that the leaf vertex on the branch connecting x_j^t to the hub h is $x_j^{s(j)}$.

Let $x_j^{s_j}$ be serviced by agent i. Consider allocation A' which is identical to A with the exception that i services all of $x_j^1, \cdots, x_j^{s_j}$. Clearly, the cost of i is the same as that in A, but the costs of all other agents either decrease or remain the same. As multiple agents, including i visit x_j^t, the cost of at least one agent reduces. Consequently, A' pareto dominates A. Thus, every PO allocation in our constructed graph must be SO.

Hence, any polynomial time algorithm to find an MMS and PO allocation or find an EF1 and PO allocation when one exists, would also find the corresponding SO allocations and solve 3-Partition. Thus, we have the proposition. □

3.5 Computing Fair and Efficient Solutions

We have shown that finding an allocation that satisfies a combination of fairness and efficiency requirements (or deciding if it exists) is computationally hard. In this section, we develop a recursive algorithm for computing each combination of the fairness-efficiency notions that is XP with respect to the number of agents, i.e., when the number of agents is bounded, the running time of our algorithm is polynomial.

Let us start by defining a notion of *Pareto frontier* as a set of Pareto optimal solutions and introduce Algorithm 3.2 that finds such set for every delivery instance.

Definition 3.11 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, its* Pareto frontier *is a minimal set of allocations $\mathcal{F}$ such that for every* PO *allocation A there exists $B \in \mathcal{F}$ and permutation of agents θ such that $c(A_i) = c(B_{\theta(i)})$, for every $i \in \mathcal{A}$.*

Algorithm 3.2 Overview. Throughout the algorithm, we keep allocations in the list $\mathcal{F}$, each allocation sorted in non-increasing cost order. First, we initialize it with just one empty allocation. Then, we look at vertices directly connected to the hub. For each, say u, we run our algorithm on a smaller instance where the graph is just the branch outgoing from h that u is on, and u is the hub. In each allocation in the output, $\mathcal{F}'$, we add u to the bundle of the first agent. Finally, we combine these allocations, with allocations on previously handled branches using an auxiliary procedure (Algorithm 3.3). Broadly, it looks at all possible combinations of allocations in both lists, and keeps only the ones that are not Pareto dominated by any other.

Algorithm 3.2: FindFrontierPO($n, \mathcal{G}, h$)

Input: number of agents n, graph $\mathcal{G}$ rooted at h
Output: Frontier $\mathcal{F}$

$\mathcal{F} := [(\emptyset, \ldots,)]$;
for $u \in$ *children of* h **do**
  $T_u :=$ subtree rooted in u;
  $\mathcal{F}' :=$ FindParetoFrontier(n, T_u, u);
  for $A \in \mathcal{F}'$ **do**
    add u to A_1 ;
  $\mathcal{F} =$ CombineFrontiers($\mathcal{F}, \mathcal{F}'$) ▷ Algorithm 3.3;

Example 3.3 *We run our algorithm on an instance with 2 agents and the graph from Figure 3.1. First, we run our algorithm on two smaller instances: one where the graph is just vertex a and one where it is the branch containing vertices b, c, d, e, f and g. Vertex a is a leaf, so from*

lines 3–5, we get just one allocation, i.e., $\mathcal{F} = \{(\{a\}, \emptyset)\}$. *When* b *is the hub, we obtain two allocations: either one agent services* g *along with all vertices on the way and the other agent services* c*, or one agent services everything.*

Thus, $\mathcal{F}' = \{(\{b, d, e, f, g\}, \{c\}), (\{b, c, d, e, f, g\}, \emptyset)\}$. *Finally, we combine* $\mathcal{F}$ *with* $\mathcal{F}'$. *We consider all four possible combinations. However, one of them,* $(\{a, b, d, e, f, g\}, \{c\})$ *is Pareto dominated by another,* $(\{b, c, d, e, f, g\}, \{a\})$ *(the cost of the first agent is the same, but for the second agent it decreases by 1). In conclusion, we return three allocations:* $(\{b, d, e, f, g\}, \{a, c\})$, $(\{b, c, d, e, f, g\}, \{a\})$, *and* $(\{a, b, c, d, e, f, g\}, \emptyset)$. *We note that the first one is in fact* MMS *and* PO *allocation.*

We begin by introducing some additional notation. By Θ_n, we denote the set of all permutations on set $\mathcal{A}$. For a permutation $\theta \in \Theta_n$ and allocation $A = (A_1, \ldots, A_n)$, by $\theta(A)$ we understand allocation $(A_{\theta(1)}, \ldots, A_{\theta(n)})$, i.e., allocation where agent i receives the original bundle of agent $\theta(i)$. For two partial allocations, $A = (A_1, \ldots, A_n)$ and $B = (B_1, \ldots, B_n)$, with disjoint set of distributed vertices, i.e., $\bigcup_{i \in \mathcal{A}} A_i \cap \bigcup_{i \in \mathcal{A}} B_i = \emptyset$, by $C = A \oplus B$, we understand allocation $C = (A_1 \cup B_1, \ldots, A_n \cup B_n)$. For two allocations A and B we write $A <_{PO} B$, when A is Pareto dominated by B. We write $A \leq_{PO} B$, when it is *weakly Pareto dominated*, i.e., $c(A_i) \geq c(B_i)$ for every $i \in \mathcal{A}$. Now, we are ready to describe auxiliary Algorithm 3.3 that combines two lists of allocations, $\mathcal{F}$ and $\mathcal{F}'$.

Algorithm 3.3: CombineFrontiers($\mathcal{F}, \mathcal{F}'$)

Input: Number of agents n, Frontiers $(\mathcal{F}, \mathcal{F}')$
Output: Res

```
1 Res = [ ], empty list of allocations;
2 for A ∈ F, B ∈ F', θ ∈ Θ_n do
3     C = sort(A ⊕ θ(B)) ;
4     if there is no D ∈ Res s.t. C ≤_PO D then
5         add C to Res;
6         while there is D ∈ Res s.t. D <_PO C do
7             remove D from Res;
```

Algorithm 3.2 Overview First, we initialize empty output list Res (line 1). Then, we iterate over all possible triples (A, B, θ), where A and B are allocations from input lists $\mathcal{F}$ and $\mathcal{F}'$, respectively, and θ is a permutation of agents $\mathcal{A}$ (line 2). For each such triple, we consider allocation $C = \text{sort}(A \oplus \theta(B))$, i.e., the allocations in which agent i, receives bundle $A_i \cup B_{\theta(i)}$, sorted in non-increasing cost order. In the next step, we check if there exists an allocation D

in Res that weakly Pareto dominates C (line 4). If this is the case, we disregard C and move to the next triple. If this is not the case, then we add allocation C to the list Res (line 5) and remove all allocations D from Res that are Pareto dominated by C (lines 6–8). We note that operations in lines 4–8 can be performed more efficiently if we keep allocations in Res in a specific ordering, but since it is not necessary for our results, we do not go into details for the sake of simplicity. After considering all pairs and permutations, we return list Res (line 11).

Now, let us prove the correctness of our algorithm.

Theorem 3.5 *Given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, Algorithm 3.2 computes its Pareto frontier and runs in time $O((n+2)!m^{3n+2})$.*

Proof: We shall first establish the correctness of the algorithm and then its running time.

Correctness. Let us start by showing that the output is a Pareto frontier, which we will prove by induction on the number of edges in a graph. If there is only one edge, the graph consists of the hub, h, and one vertex connected to it, say u. Then, there is only one possible allocation (up to a permutation of agents), namely, $(\{u\}, \emptyset, \ldots, \emptyset)$, and it is PO. Observe that this is also the only allocation returned by our algorithm for such a graph. Hence, the inductive basis holds.

Now, assume that our algorithm outputs a Pareto frontier for every instance, in which the number of edges is smaller or equal to M for some $M \in \mathbb{N}$ and consider an instance $\langle \mathcal{A}, \mathcal{G}, h \rangle$ with $M+1$ edges. Take arbitrary PO allocation A. We will show that in the output of Algorithm 3.2 for this instance, $\mathcal{F}$, there exists allocation B such that $c(A_i) = c(B_{\theta(i)})$ for some permutation $\theta \in \Theta_n$.

If h has only one child, u, then observe that every agent that services some vertex in A has to visit u. Also, the agent that services u services also other vertices (otherwise giving u to an agent that services some other vertex would be a Pareto improvement). Hence, partial allocation A' obtained from A by removing u is still PO and the cost of each agent is the same in both allocations. Observe that A' is also a PO allocation in instance $\langle \mathcal{A}, T_u, u \rangle$. Let $\mathcal{F}'$ be the output of Algorithm 3.2 for instance $\langle \mathcal{A}, T_u, u \rangle$. Since T_u has M edges, from inductive assumption we know that there exists $B' \in \mathcal{F}'$ and $\theta \in \Theta_n$ such that $c(A'_i) = c(B'_{\theta(i)})$, for every $i \in \mathcal{A}$.

Let B be an allocation obtained from B' by adding u to the bundle of agent 1. Since we sort allocation so that consecutive agents have non-increasing costs, we know that $B'_1 \neq \emptyset$. Hence, cost of each agent in B is the same as in B'. Hence, $c(A_i) = c(A'_i) = c(B'_{\theta(i)}) = c(B_{\theta(i)})$. Since B is in the output of the algorithm, the induction thesis holds.

Now, assume that h has more than one child. Let us denote them as $y^1, \ldots, y^k$ and by $Y^1, \ldots, Y^k$ let us denote the respective branches outgoing from h. Let $A^1, \ldots, A^k$ be partial allocations obtained from A by restricting A to one branch from $Y^1, \ldots, Y^k$, respectively (i.e., removing all vertices not in the branch from all of the bundles). Observe that since A is PO, each allocation $A^1, \ldots, A^k$ is also PO (otherwise a Pareto improvement in A^j for some $j \in [k]$ would be a Pareto improvement also in A). With the same reasoning as in the previous paragraph, by line 6 of Algorithm 3.2 in the iteration of the loop for each child y^j, list $\mathcal{F}'$ contains an allocation B^j such that $c(A^j_i) = c(B^j_{\theta^j(i)})$ for some $\theta^j \in \Theta_n$ and every $i \in \mathcal{A}$.

Now, when we combine $\mathcal{F}$ with $\mathcal{F}'$, we consider all possible combinations of allocations in $\mathcal{F}$ and $\mathcal{F}'$ along with all possible permutations of agents. Hence, in the output of the algorithm, there will be allocation B and permutation $\theta \in \Theta_n$ such that $B_{\theta(i)} = B^1_{\theta^1(i)} \cup \cdots \cup B^k_{\theta^k(i)}$ for every $i \in \mathcal{A}$, unless there is some allocation D that weakly Pareto dominates B. Since $A^1, \ldots, A^k$ are partial allocations of separate branches and $B^1, \ldots, B^k$ as well, we have

$$c(B_{\theta(i)}) = \sum_{j \in [k]} c(B^j_{\theta^j(i)}) = \sum_{j \in [k]} c(A^j_i) = c(A_i),$$

for every $i \in \mathcal{A}$. Hence, B is PO. Thus, if there is D that weakly Pareto dominates B, then $c(D_i) = c(B_i)$ for every $i \in \mathcal{A}$. Either way, there exists an allocation in $\mathcal{F}$ that for corresponding agents gives the same costs as allocation A. Therefore, $\mathcal{F}$ is a Pareto frontier, which concludes the induction proof.

Running Time. In the rest of the proof, let us focus on showing that the running time of Algorithm 3.2 is $O((n+2)!m^{3n+1})$. To this end, recall that in lines 2–10 of auxiliary Algorithm 3.3 we consider all triples (A, B, θ), where A is an allocation in $\mathcal{F}$, B an allocation in $\mathcal{F}'$, and θ a permutation in Θ_n. Since the cost of an agent is an integer between 0 and m and at any moment we cannot have two allocation with the same cost for every agent in $\mathcal{F}$ or $\mathcal{F}'$ (because one is weakly Pareto dominated by the other), the sizes of $\mathcal{F}$ and $\mathcal{F}'$ are bounded by $(m+1)^n$. Hence, the loop in lines 2–10, will have at most $n!(m+1)^{2n}$ iterations. For each triple, we have to sort the resulting allocation, which can take time $n\log(n)$. Moreover, we may need to check whether resulting allocation C Pareto dominates or is Pareto dominated by all of the allocations already kept in Res. The size of Res is also bounded by $(m+1)^n$ and checking Pareto domination can be done in time n. All in all, the running time of Algorithm 3.3 is in $O((n+2)!m^{3n})$. Finally, observe that in Algorithm 3.2 we call Algorithm 3.3 less than m times, thus final running time is in $O((n+2)!m^{3n+1})$. □

Now, let us show how we can use Algorithm 3.2 to find allocation satisfying certain fairness and efficiency requirements (or decide if they exist).

Finally, we show how Algorithm 3.2 can be used to obtain the desired allocations.

Theorem 3.6 *There exists an XP algorithm parameterized by n, that given a delivery instance $I = \langle \mathcal{A}, \mathcal{G}, h \rangle$, computes an* MMS *and* PO *allocation, and can decide whether the instance admits* MMS *and* SO*,* EF1 *and* PO*, and* EF1 *and* SO *allocations.*

Let us split the proof into four lemmas devoted to each combination of fairness and efficiency notions. We start with MMS and PO allocations.

Lemma 3.1 *There exists an XP algorithm parameterized by n that for every delivery instance $\langle \mathcal{A}, \mathcal{G}, h \rangle$ finds an* MMS *and* PO *allocation.*

Proof: Observe that the allocation in the Pareto frontier that leximin dominates all other allocations in the frontier is a leximin optimal allocation. Since Algorithm 3.2 returns a Pareto frontier, and Pareto frontier contains at most $O(m^n)$ allocations, we can find such an allocation in XP time by Theorem 3.5. Therefore, the lemma follows from Proposition 3.3. □

Next, let us move to EF1 and PO allocations.

Lemma 3.2 *There exists an XP algorithm parameterized by n that for every delivery instance $\langle \mathcal{A}, \mathcal{G}, h \rangle$ decides if there exists an* EF1 *and* PO *allocation and finds it if it exists.*

Proof: From the proof of Theorem 3.4 and Proposition 3.7 we know that an EF1 and PO allocation is leximin optimal and the pairwise differences in the costs of agents are at most 1. Observe that it is an equivalence, i.e., leximin optimal allocation in which the pairwise differences in the costs of agents are at most 1 is EF1 and PO (PO because of leximin optimality, and EF1 because for every agent there exists a vertex that removed from a bundle of this agent reduces the cost by at least 1). By Lemma 3.1, we can find a leximin optimal allocation in XP time with respect to n. Therefore, it remains to check if the pairwise differences in the costs of agents are at most 1. If it is true, then we know that this allocation is EF1 and PO. Otherwise, we know there is no EF1 and PO allocation. □

Now, let us consider MMS and SO allocations.

Lemma 3.3 *There exists an XP algorithm parameterized by n that for every delivery instance $\langle \mathcal{A}, \mathcal{G}, h \rangle$ decides if there exists an* MMS *and* SO *allocation and finds it if it exists.*

Proof: For MMS and SO, observe that an SO allocation is also necessarily a PO allocation. Having the Pareto frontier from Theorem 3.5 and MMS value from Lemma 3.1, we can simply check all allocation in the frontier if they satisfy MMS and SO. □

Finally, we focus on EF1 and SO allocations.

Lemma 3.4 *There exists an XP algorithm parameterized by n that for every delivery instance $\langle \mathcal{A}, \mathcal{G}, h \rangle$ decides if there exists an* EF1 *and* SO *allocation and finds it if it exists.*

Proof: For EF1 and SO, observe that Theorem 3.4 also implies that EF1 and SO allocation is MMS. From Lemma 3.3 we know that we can find all MMS and SO allocations in the Pareto frontier in XP time with respect to n. Hence, we can check if any one of them satisfies also EF1 and this will give us the solution. □

From Lemmas 3.1, 3.2, 3.3 and 3.4 we have the result.

3.6 Experiments

We now present our experimental results complementing the theoretical findings presented earlier in the chapter. In each experiment, we generated random trees based on Prüfer sequences [99] using NetworkX Python library [62]. For each experiment and a graph size, we sampled 1,000 trees.

Experiment 1. Running Time of Algorithm 3.2

We run Algorithm 3.2 for graphs of sizes $10, 20, \ldots, 100$ and every number of agents from 2 to 6. The running times are reported in Figure 3.4 (the running time for 2 agents is not reported as it would be indistinguishable from the running times for 3 agents in the picture). The exponent in the running time of our algorithm for $n = 6$ agents is significant. Consequently, we see the sharp increase in the running time with growing graph size in this case.

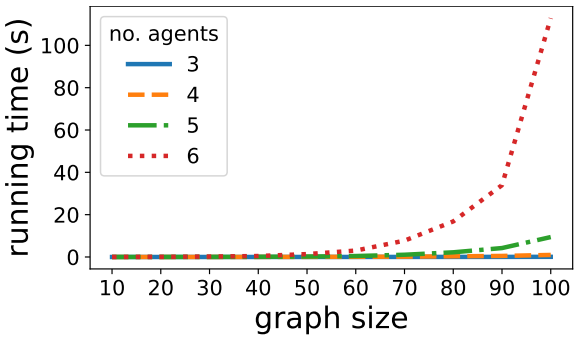
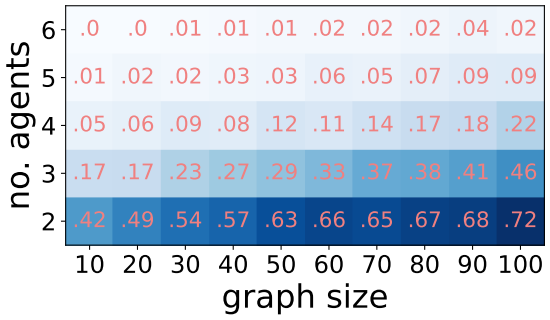


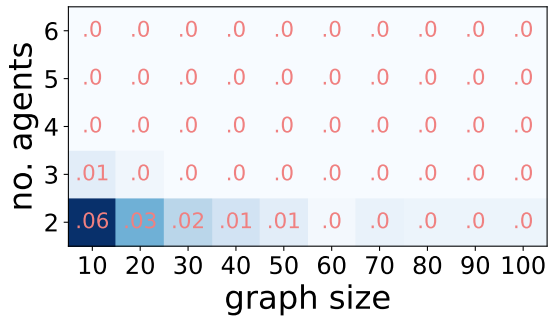
Figure 3.4: Running Time of Algorithm 3.2

Experiment 2. Existence of Fair and Efficient allocations

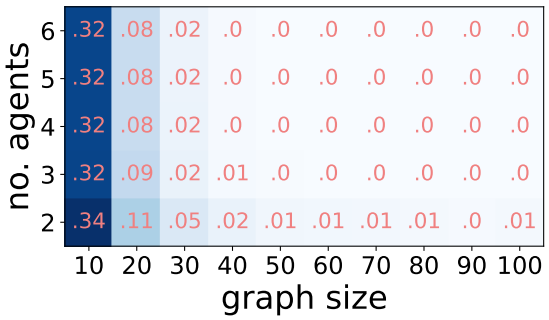
We next checked how often the fairness and efficiency combinations that need not always exist, actually exist. Recall that an MMS and PO solution always exists. Hence, we look at the probability of existence of the remaining combinations. That is, of the trees generated, we looked at the fraction of trees that for which the required fairness and efficiency combinations are satisfied. To this end, we generated trees of sizes $10, 20, \ldots, 100$ and for each tree we run Algorithm 3.2 with the number of agents varying between 2 to 6. Based on the output, we checked for the number of trees that admitted an allocation with the required combinations.



(a) EF1 and PO allocations



(b) EF1 and SO allocations



(c) MMS and SO allocations

Figure 3.5: Probability of existence of fair and efficient allocations

EF1 and PO allocation existence. First, we checked how often there exists an EF1 and PO allocation. As shown in Figure 3.5, the probability of finding an EF1 and PO allocation

increases steadily when we increase the size of the graph, but drops sharply when we increase the number of agents. Intuitively, on larger graphs we have more flexibility in how we fairly split the vertices in a PO way. However, when there are more agents, it may still be difficult to satisfy fairness for each agent.

EF1 and SO allocation existence. The results for EF1 and SO allocations are presented in Figure 3.5b. There, we see a sharp decrease in probability with the increase in either number of agents or the size of a graph. For the former it is clear, as with larger number of agents it is difficult to be fair to all of them. For the size of a graph, recall Corollary 3.1 in which we have shown that EF1 and SO allocation exists only if the hub is in the center of a tree. Moreover, the tree always contains one or two vertices. Hence, with the increase in the size, the probability that the hub will be in the center decreases.

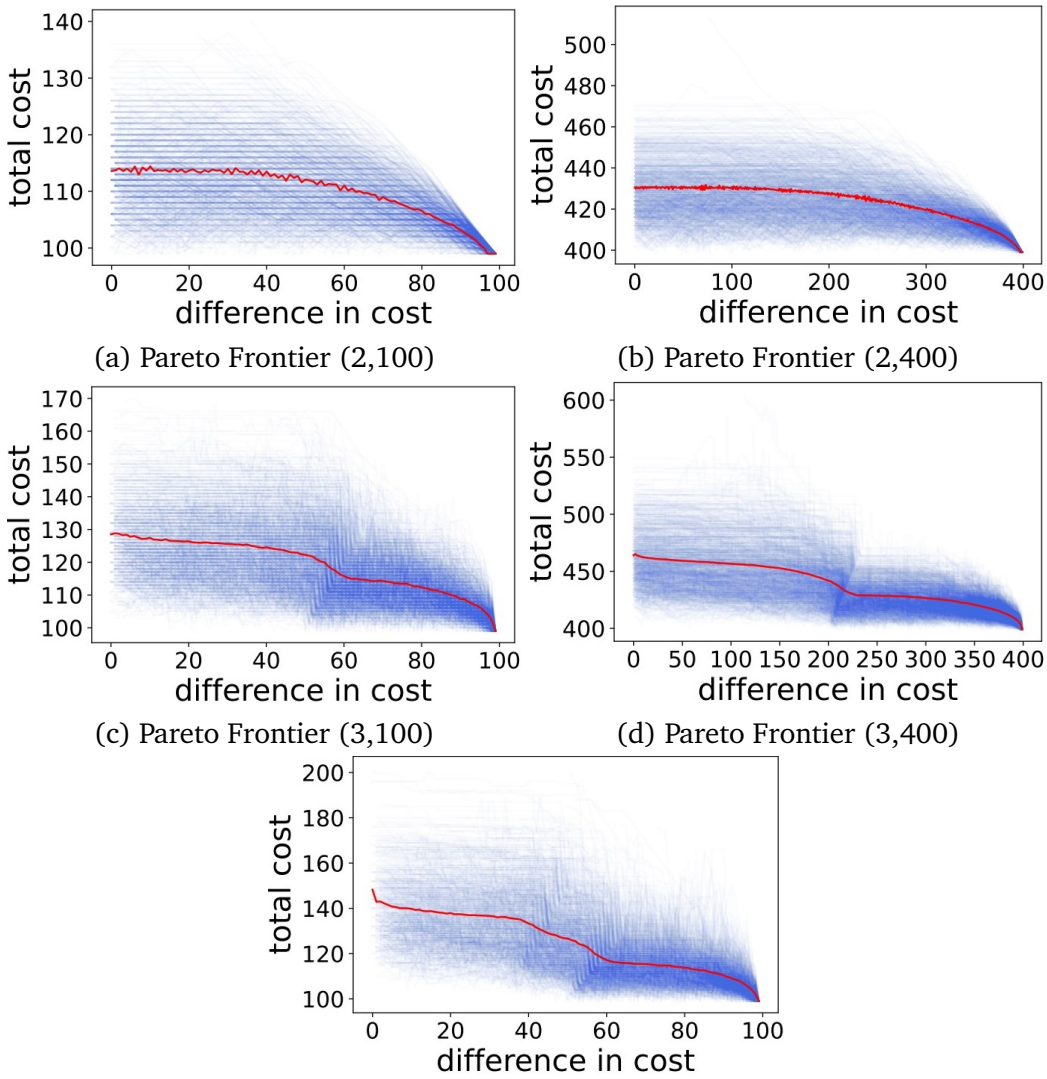
MMS and SO allocations existence. The results for MMS and SO allocations are presented in Figure 3.5c. As can be seen in the picture, the probability does not vary much between different numbers of agents (especially for small graphs). A plausible explanation for that phenomenon is that for MMS we only have to care to not be unfair to the worst off agent. If we have a small graph and a lot of agents, then probably some of them will not be assigned to any vertex either way. However, the number of such agents does not impact whether an allocation is MMS or not. Hence, the visible effect.

Experiment 3. Pareto Frontiers

In this experiment, we analyze the trade-off between fairness and the total cost of agents in singular Pareto frontiers. To this end, we conduct multiple simulations, with graphs of size either 100 or 400 and the number of agents between 2 and 4. For 4 agents, we only run the experiment on graphs of size 100. For each one of 1000 trees generated (for each graph size and number of agents pair), we look at each allocation in the Pareto frontier and report the total cost of both agents on the y-axis and the difference in costs of agents on the x-axis. Then, we connect all such points for allocations in one Pareto frontier to form a partially transparent blue line. By superimposition of all 1000 of such blue lines, we obtain a general view on the distribution of the Pareto frontiers. With the thick red line we denote the average total cost, for each difference in costs. We see that particular Pareto frontiers can behave very differently, but the general pattern is quite strong.

Two Agents

With a graph size of 400 and 2 agents, shown in Figure 3.6b the total cost does not vary much when the difference in costs is between 0 and 250, however it is much steeper for the larger



(a) Pareto Frontier (2,100)

(b) Pareto Frontier (2,400)

(c) Pareto Frontier (3,100)

(d) Pareto Frontier (3,400)

(e) Pareto Frontier (4,100)

Figure 3.6: Under "Pareto frontier (n, m)", we see the distribution of Pareto frontiers for n agents with graphs of size m.

differences. These findings imply that it is usually not effective to focus on partial fairness as the additional total cost that we incur by guarantying complete fairness instead of partial is not significantly large.

The plot for the size 100 and 2 agents, shown in Figure 3.6a looks very similar to the one presented in Figure 3.6b. Again, we can say, that the total cost for agents increases sharply when we decrease the difference in costs of agents from 400, but the further we go, this increase is slower.

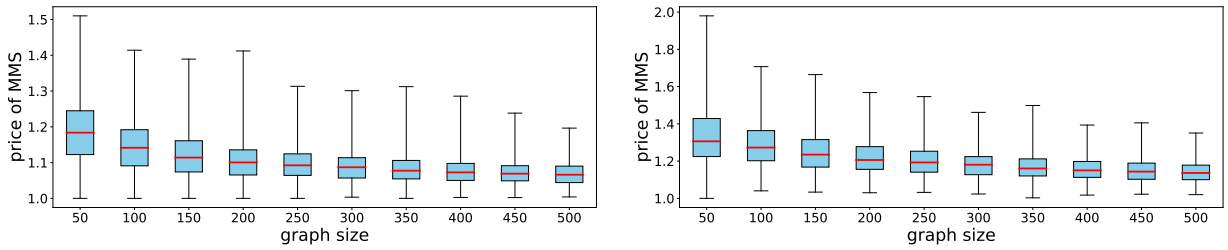
Three Agents

For 3 agents and both graph sizes, shown in Figures 3.6c and 3.6d, the plots look similar to these for 2 agents in their right-hand side part. However, a little to the right from the middle of the picture, we see a sudden sharp increase in the total cost that later also flattens.

We offer the following interpretation of this fact: in the rightmost part of the plot we begin with an allocation where the first agent is serving all of the nodes, which gives us the minimal total cost, but also the maximal difference between the costs of two agents. Then, when we want to decrease the difference, we have to take some of the vertices served by the first agent, and give it to some other agent. However, as only the maximal difference between the agents is important to us, we can, for now, split the vertices only between the first two agents, which is easier (i.e., results in a smaller total cost). However, when we reach the difference in cost that is around half of the graph size, this is no longer possible. To decrease the difference further, we have to start assigning vertices to the third agent, which brings additional cost. Thus, the increase in the total cost in the left-hand side half of the plot.

Four Agents

The situation for 4 agents, shown in Figures 3.6e, is similar to that of 3 agents, but here after the first increase in a bit more than a half of the plot, where we start assigning vertices to the third agent, we see a second increase in a bit more than a third of the plot, where we start assigning vertices to the forth agent.



(a) Price of MMS with 2 agents

(b) Price of MMS with 3 agents

Figure 3.7: Distribution and median prices of fairness

Experiment 4. Price of fairness.

In our final experiment, we analyze price of fairness of MMS allocations. To this end, we generated trees of sizes $50, 100, 150, \ldots, 500$ for two and three agents and ran Algorithm 3.2. We define the price of fairness as the ratio of the minimum total cost of agents in an MMS allocation to the minimum total cost in any allocation (i.e., the number of edges in a graph).

The results are reported in boxplots in Figure 3.7a and 3.7b. The peak median cost for 2 agents is around 1.2 and for 3 agents is around 1.3. The increase in the price of fairness is consistent with the observations in the previous experiment. As can be seen, in both cases, the median price gradually decreases with the increase of the graph size. These results suggest that as the size of the instance grows, the efficiency loss due to MMS becomes negligible in most cases (at least for a small number of agents).

3.7 Conclusions and Future Work

We introduced a novel problem of fair distribution of delivery orders on tree graphs. We provided a comprehensive characterization of the space of instances that admit fair (EF1 or MMS) and efficient (SO or PO) allocations. We showed that these problems are computationally hard and developed an XP algorithm parameterized by the number of agents for each combination of fairness-efficiency notions. We complemented this with detailed experimentation.

Our work paves the way for future research on developing approximation schemes or perhaps algorithms parameterized by graph characteristics (e.g., maximum degree or diameter) in this domain. Another intriguing direction is generalizing our fair delivery framework to account for cyclic graphs, heterogeneous cost functions, or capacity constraints on delivery orders assigned to each agent. A detailed discussion is provided in Chapter 7.1.

Chapter 4

Repeatedly Matching Items to Agents Fairly and Efficiently

In this chapter, we consider a novel setting where items are matched to the same set of agents repeatedly over multiple rounds. Each agent gets exactly one item per round, which brings interesting challenges to finding efficient and/or fair *repeated matchings*. A particular feature of our model is that the value of an agent for an item in a particular round depends on the number of rounds in which the item has been used by the agent in the past. We present a set of positive and negative results about the efficiency and fairness of repeated matchings. For efficiency, we pursue social welfare maximization. In general, it is NP-Hard to find a social welfare maximizing repeated matching. However, when values are monotone in time, either monotone increasing or decreasing, the problem becomes tractable. For fairness, when all the items are goods, an adaptation of envy-freeness up to one good (EF1) will be satisfied under certain conditions. Furthermore, it is intractable to achieve fairness and (approximate) efficiency simultaneously, even under conditions when they are achievable separately. For mixed items, which can be goods for some agents and chores for others, we propose and study a new notion of fairness that we call *swap envy-freeness* (swapEF).

4.1 Introduction

The standard fair division setting involves a set of items and agents who have values for these items. The objective is to compute an allocation which gives each item to a single agent so that some notion of fairness is satisfied. Prior work has typically explored various settings

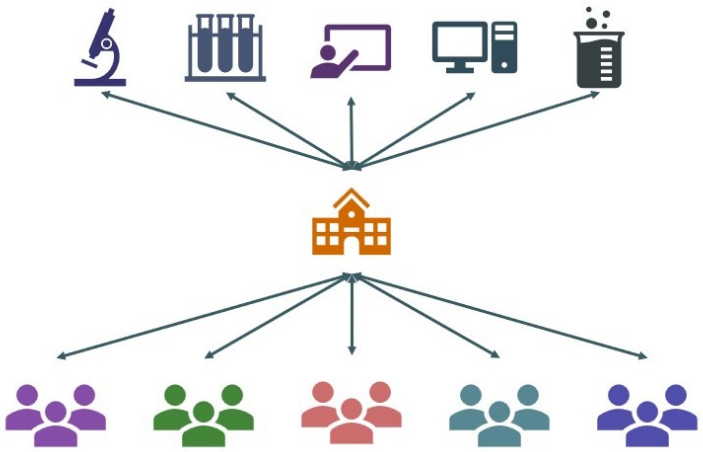


Figure 4.1: Matching research groups to university resources

where agents' allocations do not change with time. Typically, the number of items allocated to an agent is not explicitly restricted, with the exception of some very recent work [27, 49, 59]. However, sometimes in practice, the same set of items must be allocated to the same set of agents repeatedly. More crucially, another feature that distinguishes such scenarios from the standard setting is that the value of an agent for an item changes over time and typically depends on how many times the agent has received the item in the past. This can make solutions that were fair when the agents were allocated the items once, no longer fair when these agents get these items every time.

Consider the following example, there are different research labs that all need access to several expensive research facilities in a university. The access of the labs to the facilities must be fairly coordinated/scheduled throughout the semester or year. This is a fair division problem with the labs and the facilities playing the role of the agents and the items, respectively. To be fair among labs and efficient overall, such a scheduling should take into account the values the labs have for facilities, which typically change over time. For instance, during the first few weeks of access to a facility, the researchers in a lab may need time to learn how to operate it. During that time, the value the lab gets by accessing a facility can be very low, even negative. As the researchers gain more experience, their research output increases, and so does the lab's value for the facility. Once the researchers have run their intended experiments, the lab's value for the facility decreases again until the next experiment.

To capture such situations, we introduce a new model of *repeated matchings* with n agents who must be matched with exactly one of n items in each of T rounds, repeatedly. An important novelty of our model is that valuations are *history-dependent*: the value an agent has for an item in a round depends on how many times the agent has used the item in

previous rounds. Such valuations cause many challenges to achieving efficiency and fairness. We use *social welfare* (the total value of the agents from the items they get in all rounds) to as the index of the efficiency of repeated matchings.

We also use relaxations of *envy-freeness* as fairness concepts. We adapt the well-known *envy-freeness up to one item* (EF1) and use it when all valuations are non-negative (i.e., when items are *goods*). A repeated matching is EF1 if the value of every agent i for her bundle is at least as high as her value for the bundle of any other agent j after removing the last *copy* of an item from j's bundle. We observe that EF1 is not suitable when valuations can be positive or negative (i.e., when items are *mixed*), and introduce the notion of *swap envy-freeness* (swapEF) to assess fairness of repeated matchings for mixed items.

4.1.1 Technical Contributions of this Chapter

More specifically, the technical contributions of this study are as follows. We prove that the problem of computing a repeated matching with maximum social welfare is NP-hard, even when $T = 3$. Our hardness reduction defines instances with non-monotone valuations. Surprisingly, we find that the problem becomes solvable in polynomial time when the valuations are monotone. This is when the value an agent has for an item can only decrease or increase, but not both, in terms of the number of rounds the agent had been matched to the item in the past. For the case of monotone non-increasing valuations, earlier work on b-matchings can be leveraged to find the optimal solution. When the valuations are monotone non-decreasing, we find a neat reduction to the case of time-constant valuations which can be solved efficiently. In Appendix A.2, we describe how to find a social welfare maximizing repeated allocation.

We begin our fairness considerations with EF1 as a fairness concept. We find that under identical valuations, EF1 repeated matchings always exist and can be found in polynomial time. Furthermore, we show that any instance with general valuations and $T \bmod n \in \{0, 1, 2, n-1\}$ (i.e., including all instances with at most four agents/items) has an EF1 repeated matching, which can be computed efficiently. We establish that, unfortunately, EF1 is not compatible with social welfare maximization and even approximating the maximum social welfare over EF1 repeated matchings is NP-hard. This holds even for settings where EF1 solutions and social welfare maximizing solutions can be found in polynomial time separately. These interactions are illustrated in Figure 4.2.

Moreover, at a conceptual level, we propose and study a new fairness notion called swap envy-freeness (swapEF). Here, we find that under identical valuations, swapEF repeated matchings can be found using the same algorithm as used for EF1. Furthermore, we show

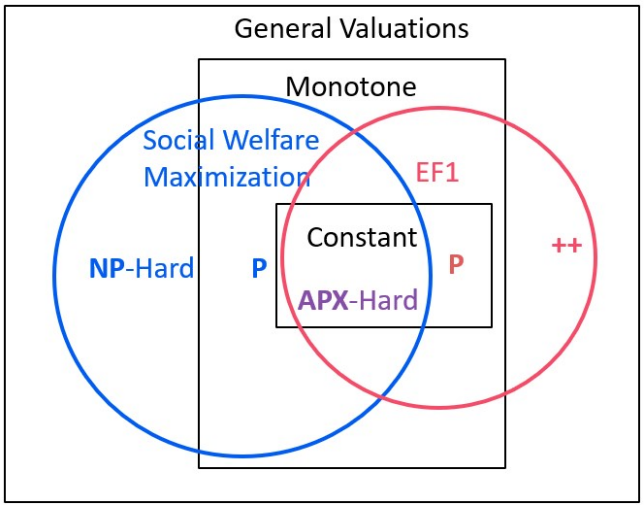


Figure 4.2: Fairness and Efficiency Interactions under Repeated Matchings

that swapEF repeated matchings always exist and can be computed efficiently on instances with $T \bmod n \in \{0, 1, 2, n-2, n-1\}$ and mixed items (i.e., including all instances with at most five agents/items). We conclude the chapter by showing the non-existence of EFX solutions in our setting. Our hardness results are proved on instances with goods. Our positive results besides those for EF1, apply to instances with mixed items.

We begin with setting up the notation and relevant definitions in Section 4.2. In Section 4.3 we focus on maximizing social welfare. Here, we give our hardness result for maximizing social welfare in general and polynomial-time algorithms for monotone valuations. We explore settings under which we can satisfy EF1 and algorithms that find EF1 solution in Section 4.4. In Section 4.5.1, we find that even in settings where EF1 repeated matchings can be found in polynomial time, maximizing social welfare over the space of EF1 repeated matchings is intractable. We devote Section 4.6 to the study of swap envy-freeness.

4.1.2 Prior Relevant Work

The standard fair allocation setting assumes that each item is given to exactly one agent with no explicit cap on the number of items one agent can get. Biswas and Barman [27] consider a fair division model where the items are partitioned into categories and there are cardinality constraints on how many items an agent can be allocated from each category. They show how to compute an EF1 allocation by extending the envy-cycle elimination algorithm of Lipton et al. [82]. Even though cardinality constraints can imply allocations which look like repeated matchings, our history-dependent valuations cannot be expressed by their model. Another extension of the standard setting is considered by Gafni et al. [51] where each item

may have multiple copies. They study relaxations of envy-freeness with mixed items, in a model where each agent can get at most one item copy. Some work has also been done on envy-freeness and its relaxations, in the roommate matching model where agents get value from their rooms as well as roommates [38, 78, 80]. Here the size of the bundle an agent gets (their room and roommates) is determined by the capacity of the room.

The concept of repeated matching has been considered before, actually using EF1 as fairness concept. However, history-dependent valuations have not. Hosseini et al. [64] look at a dynamic one-sided repeated matching model with ordinal preferences that change over time. They study strategyproofness and give a mechanism that is EF1. As the model of preferences studied is entirely different, their results are not applicable to our model. Gollapudi et al. [59] study a two-sided repeated matching setting where the agent values may change in each round, but they too, do not take into account how often the two agents have been matched in the past. In addition, due to the two-sided nature of their setting, their results are not applicable to our case.

Finally, relaxations of envy-freeness have been considered extensively in the literature. For mixed items in particular, [3] summarizes the several variations of EF1 that have been proposed in the literature and proposes new ones. Others include EFL, envy-freeness up to one less preferred good [14], EFR, envy-freeness up to a random good [43] and iEF, interim envy-freeness [37]. The setting of identical valuations has also been specifically explored, both for the existence of almost envy-free solutions [41, 96] and other fairness objectives [11, 16]. To the best of our knowledge, swap envy-freeness appears to be novel.

4.2 Notation and Preliminaries

This chapter considers instances with a set $\mathcal{A}$ of n agents and a set $\mathcal{G}$ of n items.We look at *repeated matchings*, where the items are matched to the agents in T rounds. By matching, here, we mean a one-one matching in each round. More formally, we consider instances of the form $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i\in\mathcal{A}}, T\rangle$, where T denotes the number of rounds and, for each agent $i \in \mathcal{A}$, v_i is a function from $\mathcal{G} \times [T]$ to $\mathbb{R}$, where $v_i(g, t)$ denotes the *valuation* of agent i for item g when it is matched to the item for the t^{th} time. A repeated matching $A = (A^1, ..., A^T)$ is simply a collection of matchings, with one matching A^t per each round $t \in [T]$. Furthermore, we denote by A_i the multiset (or *bundle*) which contains copies of the items to which agent $i \in \mathcal{A}$ is matched in the T rounds.

Hence, defining the bundles A_i for $i \in \mathcal{A}$ given the repeated matching A is trivial. The opposite task is also largely straightforward. Let $N(B, g)$ be the multiplicity of item g in bundle B. Given bundles of items A_i for $i \in \mathcal{A}$ with $|A_i| = T$ (i.e., each agent gets T copies

of items) and $\sum_{i \in \mathcal{A}} N(A_i, g) = T$ (i.e., T copies of each item g are allocated), a consistent repeated matching[1] for instance I is obtained as follows. We construct the bipartite multigraph $G = (\mathcal{A}, \mathcal{G}, E)$ so that the set of edges E consists of (a copy of) edge (i, g) for every (copy of) item g such that $g \in A_i$. The graph G is T-regular and, thus, by Hall's matching theorem (see 97), can be decomposed into T matchings of edges M_1, ..., M_T. These matchings correspond to a repeated matching by interpreting the edge (i, g) in matching M_t as the assignment of item g to agent i in the t^{th} round.

With a slight abuse of notation, we use $v_i(B)$ to denote the *value* agent $i \in \mathcal{A}$ has when she gets the bundle B, i.e.,

$$v_i(B) = \sum_{g \in \mathcal{G}} \sum_{t=1}^{N(B,g)} v_i(g, t).$$

Hence, for a repeated matching A, $v_i(A_i)$ is the total value from each item copy agent i receives in all rounds. The *social welfare* of A is simply the sum of the agents' values for their bundle, i.e., $SW(A) = \sum_{i \in \mathcal{A}} v_i(A_i)$.

Valuations

We shall look at specific types of valuations under which we will try to find efficient and/or fair repeated matchings. A well-motivated setting is that of *identical* valuations where $v_1 = v_2 = \cdots = v_n$. This assumption proves particularly useful in finding fair solutions. Another important class of valuation functions is that of *monotone* valuations.

Definition 4.1 (Monotone Valuations) *The valuation function v_i is monotone non-increasing (respectively, monotone non-decreasing) if for every item $g \in \mathcal{G}$, and $t \in [T-1]$, we have that $v_i(g, t) \geq v_i(g, t+1)$ (respectively, $v_i(g, t) \leq v_i(g, t+1)$).*

These two classes of valuation functions intersect in the class of *constant* valuations.

Definition 4.2 (Constant Valuations) *Valuation function v_i is said to be constant if for every item $g \in \mathcal{G}$, we have that $v_i(g, 1) = v_i(g, 2) = \cdots = v_i(g, T) = v_i(g)$.*

We extend EF1 to repeated matchings as follows.

Definition 4.3 (EF1) *A repeated matching A is* EF1 *if for every pair of agents $i, j \in \mathcal{A}$, there exists an item $g \in \mathcal{G}$ such that $v_i(A_i) \geq v_i(A_j \setminus \{g\})$.*

[1] We remark that this repeated matching is not unique. However, this does not affect the values of each agent for her bundle and the bundle of any other agent, which are the same in all different consistent repeated matchings.

We remark that the operation $A_j \setminus \{g\}$ removes one copy of item g from the bundle A_j if g belongs to A_j and leaves A_j intact otherwise.

We refer to the items as *goods* on instances where all valuations are non-negative, i.e., when $v_i(g,t) \geq 0$ for every $i \in \mathcal{A}$, $g \in \mathcal{G}$, and $t \in [T]$. When there are no restrictions on the valuations, we refer to the items as *mixed*.

4.2.1 Overview of Main Results

We now give a formal overview of the work presented in this chapter. We first look at social welfare maximization. In general, this is NP-Hard.

Theorem 4.1 *Given a repeated matching instance, computing a repeated matching of maximum social welfare is NP-hard.*

When the valuations are monotone non-increasing, prior work on b-matchings can be used to maximize social welfare. For the case of monotone non-decreasing valuations we give a neat reduction to constant valuations to show that the problem is tractable.

Theorem 4.2 *Given a repeated instance with monotone non-decreasing valuations, a repeated matching of maximum social welfare can be computed in polynomial time.*

For fairness, we first consider identical valuations and give an algorithm to find an EF1 repeated matching.

Theorem 4.3 *Given a repeated matching instance with identical and non-negative valuations, an* EF1 *repeated matching exists and can be computed in polynomial time.*

We find that we can extend the ideas used in the case of identical valuations to more general settings.

Theorem 4.4 *Given a repeated matching instance I with n agents/goods and T rounds such that $T \bmod n \in \{0, 1, 2, n-1\}$, an* EF1 *repeated matching exists and can be computed in polynomial time for non-negative valuations.*

We then look at maximizing for social welfare over the space of EF1 repeated matchings. We find that this is APX-Hard, even in the relatively more straightforward setting of constant valuations.

Theorem 4.5 *For every constant $\epsilon > 0$, approximating the maximum social welfare of* EF1 *repeated matchings on instances with n agents/goods and T rounds within a factor of $O\left(\min\{n^{1/3-\epsilon}, T^{1-\epsilon}\}\right)$ is NP-hard.*

We then consider an alternate notion of fairness that we show exists in more general settings than EF1.

Lemma 4.1 *Given a repeated matching instance $I = \langle \mathcal{A}, \mathcal{G}, v, T \rangle$ with identical valuations, the repeated matching returned by Algorithm 4.1 is* swapEF.

Theorem 4.6 *Given a repeated matching instance I with mixed items, n agents, and T rounds such that $T \bmod n \in \{0, 1, 2, n-2, n-1\}$, a* swapEF *repeated matching exists and can be computed in polynomial time.*

4.3 Maximizing Social Welfare

We first study the complexity of the problem of computing a repeated matching of maximum social welfare. Notice that if $T = 1$, this task can be easily done by computing a maximum-weight perfect matching in the complete bipartite graph $G = (\mathcal{A}, \mathcal{G}, \mathcal{A} \times \mathcal{G})$, in which edge (i, g) has weight $v_i(g, 1)$. For $T > 1$, an approach that seems natural computes gradually a maximum-weight perfect matching for each round, taking into account the matching decisions in previous rounds.

For example, consider the instance with three agents and two rounds (i.e., $n = 3$, $\mathcal{A} = \{1, 2, 3\}$, $\mathcal{G} = \{g_1, g_2, g_3\}$, and $T = 2$). The agent valuations are as follows: $v_1(g_2, 1) = v_1(g_3, 1) = 1 - \epsilon$ (for small but strictly positive ϵ), $v_2(g_2, 1) = v_3(g_3, 1) = 1$, while all other valuations are 0. We can show that a greedy approach of sequentially taking maximum weight matchings for each of T rounds will not maximize social welfare.

A maximum-weight perfect matching on the complete bipartite graph $G = (\mathcal{A}, \mathcal{G}, \mathcal{A} \times \mathcal{G})$ with weight $v_i(g, 1)$ on edge (i, g) assigns item g_i to agent i in the first round; this gives value 1 to agents 2 and 3. Then, the natural way to compute the matching of the second round is to compute a maximum-weight perfect matching in the complete bipartite graph $G = (\mathcal{A}, \mathcal{G}, \mathcal{A} \times \mathcal{G})$ with weight $v_i(g_i, 2)$ to edge (i, g_i) (because agent i already uses item i in the first round) and weight $v_i(g, 1)$ to edge (i, g) for $g \neq g_i$. In this way, the matching of the second round will give value of $1 - \epsilon$ to agent 1 only, by matching her to either item g_2 or item g_3. Thus, the social welfare is $3 - \epsilon$.

In contrast, consider the repeated matching in which the first-round matching assigns item g_2 to agent 1, item g_1 to agent 2, and item g_3 to agent 3, and the second-round matching assigns item g_3 to agent 1, item g_2 to agent 2, and item g_1 to agent 3. Agent 1 gets value $1 - \epsilon$ in both rounds, agent 2 gets value 1 in the second round, and agent 3 gets value 1 in the first round. Hence, the social welfare is now $4 - 2\epsilon$.

This example demonstrates that computing a repeated matching of maximum social welfare can be a challenging task. Actually, as our first result indicates, the problem is computationally hard.

4.3.1 Intractability of Social Welfare Maximization in General

Theorem 4.1 *Given a repeated matching instance, computing a repeated matching of maximum social welfare is NP-hard.*

Proof: We present a polynomial-time reduction from exact 3-cover (X3C). An instance of X3C consists of a universe $U = [3q]$ of elements and a collection $\mathcal{S}$ of m sets $S_1, S_2, ..., S_m$, containing three elements of U each. Deciding whether there are q mutually disjoint sets in $\mathcal{S}$ is a well-known NP-hard problem [53].

Setup. We construct an instance $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i \in \mathcal{A}}, T \rangle$ with $T = 3$, and $n = m + 3q$ agents/items. The set of agents $\mathcal{A}$ has a *set agent* i for every $i \in [m]$ and an *element agent* $m + i$ for every element i of U. The set of items $\mathcal{G}$ has an *element item* for every element $i \in [U]$, $m - q$ *space-filling* items $3q + 1, ..., m + 2q$, and q *dummy items* $m + 2q + 1, ..., m + 3q$. The valuations are as follows:

- For every $i \in [m]$ and every element $g \in S_i$, the set agent i has value $v_i(g, 1) = 1$ for the first copy of element item g.
- For every $i \in [m]$ and every $g = 3q + 1, ..., m + 2q$, the set agent i has value $v_i(g, 3) = 3$ for the third copy of the space-filling item g.
- For every $i \in [3q]$, the element agent $m + i$ has value $v_{m+i}(i, 2) = 3$ for the second copy of the element item i.
- All other valuations are 0 (including the valuation of any agent for a dummy item).

Maximizing social welfare using exact cover. We claim that there are q mutually disjoint sets in $\mathcal{S}$ if and only if there is a repeated matching of social welfare $3m + 9q$ in I. We begin by presenting a repeated matching of social welfare $3m + 9q$ when $\mathcal{S}$ has a subcollection X of q disjoint sets. For every i such that $S_i \in X$, the set agent i gets one copy of each element item corresponding to an element $g \in S_i$. Agent i gets a value of 3 in this way. For every i such that $S_i \notin X$ (notice that there are exactly $m - q$ such i's), the set agent i gets three copies of a distinct space-filling item. Hence, the set agents who do not get element items have value 3, too. For $i \in [3q]$, the element agent $m + i$ gets two copies of the element item i. Again, the element agents have all value 3. Each of the $3q$ element agents gets one distinct copy of one of the q dummy items; these do not contribute to the social welfare.

Finding exact cover using a maximum social welfare matching. Now consider a repeated matching on instance I that has social welfare $3m + 9q$. This means that each of the $m - q$ space-filling items gives value 3 to the agents while each of the $3q$ element items gives total value 4. Notice that these are the maximum contributions from each item to the social welfare. The only way that each space-filling item gives a value of 3 to the agents is when all its three copies are given to the same set agent. Hence, $m - q$ of the set agents have three copies of a space-filling item each. Also, the only way for an element item g to give value 4 to the agents is when two of its copies are given to the element agent $m + g$ and another copy is given to a set agent i such that $g \in S_i$. Hence, every set agent i who does not include a space-filling item contains a single copy of each of the three items corresponding to the elements in S_i which is not used in any other set agent. Hence, the union of the q sets corresponding to these set agents includes all elements of U. □

4.3.2 Tractability under Monotone Valuations

Fortunately, the problem of computing a repeated matching of maximum social welfare can be solved in polynomial time for monotone valuations, even when the items are mixed. Notice that the instance in the example given at the beginning of Section 4.3 belongs to the category of monotone non-increasing valuations.

4.3.2.1 Monotone Non-Increasing Valuations

For this particular case, well-known results on b-matchings can be used to find a social welfare maximizing repeated matching. In the following, we briefly explain how; recall that a b-matching in a bipartite graph is just a subset of the edges that includes at most b edges that are incident to any given node. Gabow and Tarjan [50] show how to compute a maximum-weight b-matching on input an edge-weighted bipartite multigraph in time that is polynomial in b, the size of the graph, and the number of bits required to represent the edge-weights.

Given a repeated matching instance $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i \in \mathcal{A}}, T \rangle$ where each $\{v_i\}_{i \in \mathcal{A}}$ is monotone non-increasing, construct the bipartite multigraph graph $G = (\mathcal{A}, \mathcal{G}, E)$ where E consists of T copies of edge (i, g) for each $i \in \mathcal{A}$ and each $g \in \mathcal{G}$. For each $t \in [T]$, $i \in \mathcal{A}$ and $g \in \mathcal{G}$, we set the edge weight of the t^{th} copy of edge (i, g) to $v_i(g, t)$. Now, since the v_is are monotone non-increasing, we can assume that a maximum-weight T-matching in G has the following *consecutive edge copies* property: if it contains k copies of an edge (i, g), these are the first k copies of weights $v_i(g, 1)$, ..., $v_i(g, k)$. Notice that, if this is not the case, we can redistribute the edge copies of (i, g) between agents appropriately without violating weight maximality. Now, a maximum-weight T-matching M in G naturally defines a repeated matching A_M in

I, where each i is matched to each g as many times as the number of copies of edge (i, g) M contains. Furthermore, the social welfare of A_M is equal to the weight of M and can be seen to be optimal. The reason is that any repeated matching corresponds to a T-matching with the consecutive edge copies property.

In Appendix A.1, we present an alternative approach for finding social welfare maximizing repeated matchings for instances with monotone non-increasing valuations. We use an integer linear program and use an LP solver to compute an extreme solution of the LP relaxation, which, as we show, is guaranteed to be integral.

4.3.2.2 Monotone Non-Decreasing Valuations

Neither b-matchings nor our linear programming-based approach can be used when all the valuation functions are monotone non-decreasing. Somewhat surprisingly, it suffices to resort to an even simpler ordinary matching computation in this case.

Social Welfare Maximization under Constant Valuations We remark that, on repeated matching instances with constant valuations, there is always a repeated matching of maximum social welfare in which every agent gets the same item in all rounds. To see why, consider any repeated matching A and let t be that round in which the total value the agents get from the items they get in matching A^t is maximum, that is choose

$$t \in \operatorname*{argmax}_{t'=1,\cdots T} SW(A^{t'}).$$

Then, the repeated matching which uses matching A^t in all rounds has at least as high social welfare with A. Hence, a straightforward maximum-weight matching computation can be used to compute a social welfare maximizing repeated matching for instances with constant valuations. The proof of the next theorem exploits a connection of instances with monotone non-decreasing valuations and instances with constant valuations.

Theorem 4.2 *Given a repeated instance with monotone non-decreasing valuations, a repeated matching of maximum social welfare can be computed in polynomial time.*

Proof: Consider a repeated matching instance $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i\in\mathcal{A}}, T\rangle$ with monotone non-decreasing valuations. For each agent $i \in \mathcal{A}$, we construct the constant valuation function v_i^c with $v_i^c(g) = \frac{1}{T}\sum_{t=1}^{T} v_i(g, t)$ for each item $g \in \mathcal{G}$. That is, the value that agent i gets from a copy of item g under valuation v_i^c is i's average value from g under v_i in T rounds.

Observe that, by the definition of the valuation v_i, $v_i(A_i) \leq v_i^c(A_i)$ for any repeated matching A and any agent $i \in \mathcal{A}$. This implies that the social welfare of A under the valuations

v_i is not higher than the social welfare under the valuations v_i^c. Hence, the maximum social welfare among all repeated matchings with respect to valuations v_i is not higher than the maximum social welfare among all repeated matchings with respect to valuations v_i^c. Furthermore, the maximum social welfare under v_i^c is achieved by a repeated matching $\widehat{A}$ that uses the same matching in all rounds.

Finally, note that $v_i(\widehat{A}_i) = \sum_{t=1}^{T} v_i(g_i, t) = v_i^c(\widehat{A}_i)$, where g_i is the item agent i gets in all rounds under $\widehat{A}$. I.e., the social welfare of $\widehat{A}$ is the same with respect to the original valuations v_i and the modified valuations v_i^c. Thus, to maximize the social welfare, it suffices to compute a single-round matching of maximum social welfare according the valuations v_i^c and repeat it for T rounds. □

4.4 Computing Fair Repeated Matchings

In this section, we focus on repeated matching instances with goods (i.e., non-negative valuations) and present algorithms that compute EF1 repeated matchings under different conditions. We first discuss how previous work by Biswas and Barman [27] can be adapted to handle repeated matching instances with constant valuations. We then consider identical valuations in Section 4.4.1 and conclude with our results for general non-negative valuations in Section 4.4.2.

Constant Valuations and Beyond

We now briefly discuss the seemingly related problem of fair division with cardinality constraints. Here we comment on whether existing results can be used to obtain EF1 repeated matchings in our setting. Biswas and Barman [27] consider an extension of the standard fair division setting where a set of items (goods) needs to be allocated to a set of agents with additive valuations for the items. The additional feature of their problem is that the set of items is partitioned into categories and each category has a cardinality constraint. The objective is now to compute an allocation of the items to the agents, in which the number of items each agent gets from each category does not exceed the cardinality constraint of that category. Biswas and Barman [27] show that allocations that satisfy such cardinality constraints and are furthermore EF1 do exist and can be computed in polynomial time.

Notice that the results of Biswas and Barman [27] can be used to compute EF1 repeated matchings for instances with *constant valuations*. Recall that constant valuations are those where the value for an item stays constant across all the copies received by the agent. Indeed, given a repeated matching instance $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i \in \mathcal{A}}, T \rangle$, it suffices to consider a fair division instance I' with the n agents in $\mathcal{A}$, T distinct items for each item g in $\mathcal{G}$, each of value $v_i(g)$

to agent i, and a cardinality constraint of T for the whole set of items. It can be easily seen that any EF1 allocation for instance I' naturally corresponds to an EF1 repeated matching for instance I and vice versa. Unfortunately, for non-constant valuations, this reduction does not work as it seems impossible to express the history-dependent valuations in our model with additive valuations for items in the model of Biswas and Barman [27].

An algorithmic idea for repeated matchings is to begin by assigning $\lfloor T/n \rfloor$ copies of each item to each agent and distribute the remaining $T \bmod n$ copies of each item so that each agent gets at most one additional copy. Can we achieve EF1 in this way for general valuations? This requires the computation of an EF1 repeated matchings on instances with $T < n$, in which each agent gets at most one copy of each item (and T copies in total). Even though additivity would not be a problem anymore, it is still not clear how to express such instances in the model of Biswas and Barman [27] using cardinality constraints defined on a *single* partition of the items only.

4.4.1 Identical Valuations

The algorithm we shall now present for repeated matching instances with identical valuations works as follows. It starts by assigning $\lfloor T/n \rfloor$ copies of each item to each agent. If $T \bmod n > 0$ (i.e., additional copies have to be assigned to the agents so that the repeated matching is correct), the algorithm works in a round robin fashion for $T \bmod n$ phases. In these phases, it uses a fixed ranking of the items according to the value $v(g, \lceil T/n \rceil)$ of their $\lceil T/n \rceil$-th copy. The ranking assigns each item a distinct integer rank(g) in $[n]$ such that rank$(g_1) <$ rank(g_2) implies that $v(g_1, \lceil T/n \rceil) \geq v(g_2, \lceil T/n \rceil)$. In each round-robin phase, the agents act according to the ordering $1, 2, ..., n$. When it is agent i's turn, she picks a copy of the lowest-rank item that is available.

The algorithm appears below as Algorithm 4.1. It has access to function rank() defined as above and uses the matrix f to store the number of copies of each item an agent gets. The final step is to call routine GenerateFromFreq() to transform f to the repeated matching A; this routine essentially implements the transformation described in Section 4.2 and is called at the final step of every algorithm we present in the chapter.

We now use Algorithm 4.1 to prove the next statement.

Theorem 4.3 *Given a repeated matching instance with identical and non-negative valuations, an* EF1 *repeated matching exists and can be computed in polynomial time.*

Proof: Algorithm 4.1 clearly runs in time polynomial in n and T. It remains to prove that it always returns an EF1 repeated matching. Consider its application to a repeated matching

Algorithm 4.1: Computing an EF1 repeated matching under identical valuations

Input: Identical Valuations Instance $I = \langle \mathcal{A}, \mathcal{G}, v, T \rangle$ with $|\mathcal{A}| = n$
Output: A repeated matching A

$f(i,g) \leftarrow \lfloor T/n \rfloor$, $\forall i \in \mathcal{A}$, $\forall g \in \mathcal{G}$;
if $T \bmod n > 0$ **then**
  $x_g \leftarrow T \bmod n$, $\forall g \in G$;
  for $t = 1$ *to* $T \bmod n$ **do**
    for $i = 1$ *to* n **do**
      $g' \leftarrow \operatorname{argmin}_{g:\, x_g > 0} \text{rank}(g)$ ▷ Give highest valued item which with unallocated copies ;
      $x_{g'} \leftarrow x_{g'} - 1$;
      $f(i,g') \leftarrow \lceil T/n \rceil$;
$A \leftarrow$ GenerateFromFreq(f);

instance $I = \langle \mathcal{A}, \mathcal{G}, v, T \rangle$, where v is non-negative. The repeated matching returned is clearly EF1 if T is an integer multiple of n; in this case, all agents get the same number of copies of all items and nobody is envious.

Otherwise, since $T \bmod n \leq n - 1$ copies of each item are available in the round-robin phases and all the remaining $T \bmod n$ copies of each item are picked in consecutive round-robin steps, no agent gets more than one copy of the same item in the round robin phases. Let $g_{i,t}$ be the item agent i gets in the round robin phase $t \in \{1, 2, ..., T \bmod n\}$. Consider two agents i and j and observe that the repeated matching A returned by Algorithm 4.1 satisfies

$$\begin{aligned} & v(A_i) - v(A_j \setminus \{g_{j,1}\}) \\ = & \sum_{t=1}^{(T \bmod n)-1} (v(g_{i,t}, \lceil T/n \rceil) - v(g_{j,t+1}, \lceil T/n \rceil)) + v(g_{i,T \bmod n}) \\ \geq & \sum_{t=1}^{(T \bmod n)-1} (v(g_{i,t}, \lceil T/n \rceil) - v(g_{j,t+1}, \lceil T/n \rceil)) \geq 0, \end{aligned}$$

as EF1 requires. The equality follows since both agents i and j get $\lfloor T/n \rfloor$ copies of each item at the beginning of the algorithm and, then, the valuation difference is due to the $\lceil T/n \rceil$-th copies of items allocated in the round-robin phases. The first inequality is due to the non-negativity of valuations. The second one follows since the item that agent i picks at the round-robin phase t has not higher rank than the item agent j picks in the next phase $t + 1$. □

4.4.2 General Valuations

We now prove that EF1 repeated matchings can be computed in polynomial time for general non-negative valuations when the number T of rounds and the number n of agents/items satisfy a particular condition.

Theorem 4.4 *Given a repeated matching instance I with n agents/goods and T rounds such that $T \bmod n \in \{0, 1, 2, n-1\}$, an* EF1 *repeated matching exists and can be computed in polynomial time for non-negative valuations.*

We prove Theorem 4.4 constructively, by defining two algorithms for the cases $T \bmod n \in \{0, 1, 2\}$ (Algorithm 4.2) and $T \bmod n = n - 1$ (Algorithm 4.3).

Algorithm 4.2 Overview. Algorithm 4.2 computes the number of copies of each item that each agent gets as follows. First, it gives to each agent $\lfloor T/n \rfloor$ copies of each item (line 1). If $T \bmod n \neq 0$, it then runs a round-robin phase (lines 2-7) and then, if $T \bmod n = 2$, it runs an additional reverse round-robin phase (lines 8-13). In the round-robin phase, the agents act according to the ordering $1, 2, ..., n$ (see the for-loop in lines 4-7). When it is agent i's turn to act, she gets the item $\widehat{g}$ (identified in line 5) for which her value for the $\lceil T/n \rceil$-th copy is maximum among the items that have not been given to agents who acted before i in the round-robin phase (the set variable P is used to identify these items).

In the reverse round-robin phase, the agents act according to the ordering $n, n-1, ..., 1$ (see the for-loop in lines 10-13). When it is agent i's turn to act, she gets the item $\widehat{g}$ (identified in line 11) for which her value for the next copy is maximum among the items that have not been given to agents who acted before i in the reverse round-robin phase. Finally, the algorithm transforms the matrix f indicating the number of copies of each item each agent gets to a repeated matching by calling routine GenerateFromFreq(). Algorithm 4.2 clearly runs in polynomial time.

Lemma 4.2 *The repeated matching $A = (A_1, ..., A_n)$ produced by Algorithm 4.2 is* EF1.

Proof: Let S denote the multiset that contains each item with multiplicity $\lfloor T/n \rfloor$. If $T \bmod n = 0$, then $A_i = S$ for every agent i and, hence, agents are not envious of each other. If $T \bmod n = 1$, the final repeated matching is obtained after the execution of the round-robin phase. Consider two agents i and j. Denoting by g_j the item agent j gets in this phase, agent i has value $v_i(A_i) \geq v_i(S) = v_i(A_j \setminus \{g_j\})$, i.e., she satisfies the EF1 condition.

If $T \bmod n = 2$, the final repeated matching is obtained after the execution of the reverse round-robin phase. Consider two agents i and j with $i < j$. Let g_i^1 and g_j^1 be the items the

Algorithm 4.2: Computing an EF1 repeated matching

Input: Instance $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i \in \mathcal{A}}, T \rangle$ with $|\mathcal{A}| = n$ and $T \bmod n \in \{0, 1, 2\}$
Output: A repeated matching A

$f(i, g) \leftarrow \lfloor T/n \rfloor, \forall i \in \mathcal{A}, \forall g \in G$;
if $T \bmod n > 0$ **then**
  $P \leftarrow \mathcal{G}$ ▷ Initialize for one round of round robin ;
  for $i = 1$ *to* n **do**
    $\widehat{g} \leftarrow \operatorname{argmax}_{g \in P} v_i(g, \lceil T/n \rceil)$;
    $f(i, \widehat{g}) \leftarrow \lceil T/n \rceil$;
    $P \leftarrow P \setminus \{\widehat{g}\}$;
if $T \bmod n = 2$ **then**
  $P \leftarrow \mathcal{G}$ ▷ Initialize for reverse order round robin ;
  for $i = n$ *to* 1 **do**
    $\widehat{g} \leftarrow \operatorname{argmax}_{g \in P} v_i(g, f(i, g) + 1)$;
    $f(i, \widehat{g}) \leftarrow f(i, \widehat{g}) + 1$;
    $P \leftarrow P \setminus \{\widehat{g}\}$;
$A \leftarrow$ GenerateFromFreq(f);

agents i and j get in the round-robin phase and g_i^2 and g_j^2 be the items they get in the reverse round-robin phase, respectively. Agent i has value

$$\begin{aligned} v_i(A_i) &\geq v_i(S) + v_i(g_i^1, \lceil T/n \rceil) \\ &\geq v_i(S) + v_i(g_j^1, \lceil T/n \rceil) = v_i(A_j \setminus \{g_j^2\}). \end{aligned}$$

The second inequality follows since agent i prefers item g_i^1 to item g_j^1 in the round-robin phase. For agent j, we distinguish between two cases. Let μ denote the multiplicity of item g_j^2 in A_j. If $g_j^1 \neq g_i^2$, we have that, in the reverse round-robin phase, agent j prefers the μ-th copy of g_j^2 to the $\lceil T/n \rceil$-th copy of g_i^2, i.e., $v_j(g_j^2, \mu) \geq v_j(g_i^2, \lceil T/n \rceil)$. Then, we have

$$\begin{aligned} v_j(A_j) &\geq v_j(S) + v_j(g_j^2, \mu) \\ &\geq v_j(S) + v_j(g_i^2, \lceil T/n \rceil) = v_j(A_i \setminus \{g_i^1\}). \end{aligned}$$

If $g_j^1 = g_i^2$, we have

$$\begin{aligned} v_j(A_j) &\geq v_j(S) + v_j(g_j^1, \lceil T/n \rceil) \\ &= v_j(S) + v_j(g_i^2, \lceil T/n \rceil) = v_j(A_i \setminus \{g_i^1\}). \end{aligned}$$

Thus, the EF1 conditions for agents i and j are satisfied. □

Algorithm 4.3 Overview. Algorithm 4.3 uses a similar structure to Algorithm 4.2. It starts by giving $\lceil T/n \rceil$ copies of each item to each agent (in line 1) and then removes the copy of a distinct item from each agent by running a round-robin phase (lines 2-6). When it is agent i's turn to act, she gets rid of a copy of the item $\widehat{g}$ (identified in line 4) for which her value for the $\lceil T/n \rceil$-th copy is minimum among the items that have not been gotten rid by agents who acted before i in the round-robin phase.

Algorithm 4.3: Computing an EF1 repeated matching

Input: Instance $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i \in \mathcal{A}}, T \rangle$ with $|\mathcal{A}| = n$ and $T \bmod n = n - 1$
Output: A repeated matching A

1 $f(i, g) \leftarrow \lceil T/n \rceil, \forall i \in \mathcal{A}, \forall g \in G$;
2 $P \leftarrow \mathcal{G}$;
3 ▷ Initialize for one round of round robin **for** $i = 1$ *to* n **do**
4 $\quad \widehat{g} \leftarrow \operatorname{argmin}_{g \in P} v_i(g, \lceil T/n \rceil)$;
5 $\quad f(i, \widehat{g}) \leftarrow f(i, \widehat{g}) - 1$;
6 $\quad P \leftarrow P \setminus \{\widehat{g}\}$;
7 $A \leftarrow$ GenerateFromFreq(f);

Lemma 4.3 *The repeated matching $A = (A_1, ..., A_n)$ produced by Algorithm 4.3 is* EF1.

Proof: Let i and j be two agents and denote by g_i and g_j the items that are removed from their bundles in the round-robin phase. We have

$$\begin{aligned} v_i(A_i) &= v_i(A_j) + v_i(g_j, \lceil T/n \rceil) - v_i(g_i, \lceil T/n \rceil) \\ &\geq v_i(A_j) - v_i(g_i, \lceil T/n \rceil) = v_i(A_j \setminus \{g_i\}) \end{aligned}$$

as desired. The last equality follows since A_j has exactly $\lceil T/n \rceil$ copies of item g_i. □

Theorem 4.4 implies the following corollary.

Corollary 4.1 *In repeated matching instances with up to four agents/goods, an* EF1 *repeated matching always exists.*

4.5 Impossibility Results for Repeated Matchings

In this section we shall explore the limitations of EF1 and another related notion of EFX for the space of repeated matchings. We first find that social welfare maximization over EF1 repeated matchings is intractable to even approximate.

4.5.1 Incompatibility of Fairness and Efficiency

In this section, we show that achieving the concepts of efficiency and fairness simultaneously is computationally intractable. In particular, we show in Theorem 4.5 below that even approximating the maximum social welfare of EF1 repeated matching is hard. Our proof is inspired by a reduction by Barman et al. [17] but is significantly more involved. Interestingly, it uses instances with constant valuations and comes in sharp contrast to achieving the two concepts separately. Recall that for such instances, an EF1 repeated matching can be computed in polynomial time by the techniques of Biswas and Barman [27] , while a polynomial time algorithm for computing social welfare maximizing repeated matchings follows by Theorem 4.2.

Theorem 4.5 *For every constant $\epsilon > 0$, approximating the maximum social welfare of* EF1 *repeated matchings on instances with n agents/goods and T rounds within a factor of $O\left(\min\{n^{1/3-\epsilon}, T^{1-\epsilon}\}\right)$ is NP-hard.*

Proof: We present a polynomial-time reduction which, given a graph $G = (V, E)$, constructs a repeated matching instance $I(G)$ in which the maximum social welfare over EF1 repeated matchings is in $[K, K+1)$ if and only if the maximum independent set in graph G has size K. Our construction leads to instances with $n \leq |V|^3$ agents/items and $T = |V|$ rounds. Then, the theorem follows by the next well-known result by Zuckerman [127].

Theorem 4.7 (Zuckerman [127]) *For every constant $\epsilon > 0$, approximating the maximum independent set of a graph $G = (V, E)$ within a factor of $|V|^{1-\epsilon}$ is NP-hard.*

Let δ be such that $0 < \delta < |V|^{-2}$. Let $G = (V, E)$ be a graph. Without loss of generality, we can assume that G has no isolated nodes, as the existence of such nodes just makes the independent set problem easier.

Setup. Given graph $G = (V, E)$, the instance $I(G)$ has $T = |V|$ rounds and $n = (2|V| + 1)|E| + 1$ agents/items. For every edge $e \in E$, $I(G)$ has $2|V| + 1$ *edge agents* identified as (e, i) for $i = 1, 2, ..., 2|V| + 1$. There is also a *special agent* s. For every node $u \in V$, there is a *node item* g_u. The instance also has $n - |V|$ *dummy items*. For edge $e = (x, y) \in E$, $i \in [2|V| + 1]$, and $t \in [T]$, the valuation of the edge agent (e, i) for the t^{th} copy of the node item g_u is $v_{e,i}(g_u, t) = \delta$ if $u = x$ or $u = y$, and $v_{e,i}(g_u, t) = 0$ otherwise. For node $u \in V$ and $t \in [T]$, the valuation of the special agent for the t^{th} copy of the node item g_u is $v_s(g_u, t) = 1$. All agents have zero valuations for the dummy items.

Constructing repeated matching from maximum independent set. Let K be the size of the maximum independent set in G. We claim that any EF1 repeated matching of $I(G)$ has social welfare less than $K + 1$. This will follow by two observations for any EF1 repeated matching A. First, for every edge e, there is some $i \in [2|V| + 1]$ such that the edge agent (e, i) has value 0. Assume that this is not true for edge $e = (x, y)$. Hence, $2|V| + 1$ copies of the node items g_x and g_y have been given to the edge agents corresponding to edge e. However, we only have $|V|$ copies of each item. Second, consider the node items the special agent gets. As for each edge $e = (x, y)$, there is some agent (e, i) who has zero value, the special agent can get at most one copy of node items g_x or g_y. As this holds for every $e \in E$, the node items that the special agent gets correspond to the nodes in an independent set in G. Hence, her value is at most K. The total value the edge agents get from the $|V|$ node items they get is at most $|V|^2 \cdot \delta < 1$. Hence, the social welfare of repeated matching A is less than $K + 1$.

Constructing maximum independent set from repeated matching. We now show that an EF1 repeated matching of social welfare in $[K, K + 1)$ does exist, when the graph G has an independent set S of size K. First, the special agent gets a single copy of node item g_x for each $x \in S$. The remaining copies of the node items are given to the edge agents in such a way that each edge agent corresponding to edge $e = (x, y)$ gets at most one copy of either g_x or g_y. This is always possible, since for every edge $e = (x, y)$, there are $2|V| + 1$ edge agents to get at most one copy of either node item g_x or node item g_y. Then, the copies of the dummy items are distributed so that each agent has exactly $|V|$ item copies. As every edge agent has at most one copy of a node item, the EF1 conditions between any two of them are satisfied. Finally, the EF1 is satisfied between any edge agent and the special agent since the special agent gets at most one item copy for which the edge agent has positive value. □

4.5.2 Impossibility of Almost Envy-freeness in Repeated Matchings

We now explore the non-existence of known relaxations of envy-freeness, namely EF1 and EFX. EFX is known to exist for identical valuations for any number of agents and for general valuations for up to 3 agents. We extend this notion to the repeated matchings setting as follows:

Definition 4.4 (EFX) *A repeated matching is said to be* EFX *if for any pair of agents i and j, for each good $g \in A_j$, $v_i(A_i) \geq v_i(A_j \setminus g)$.*

Clearly, for time-constant valuations, this simplifies to for any pair of agents i and j, for each good $g \in A_j$, $v_i(A_i) \geq v_i(A_j) - v_i(g)$. We now show that for $n = 2$, this is impossible to achieve for any odd value of $T > 1$, that is, if there exists $k \in \mathbb{Z}^+ \setminus \{0\}$ s.t. $T = 2k+1$. Consider

two identical agents who have time-constant value $2T+1$ for g_1 and 1 for g_2. Now, whenever T is odd, any repeated matching will match one agent to g_1 more than the other. Let A be an arbitrary repeated matching and agent i be the agent matched to g_1 more often. If $g_2 \notin A_i$, then $v_j(A_j) = T < (2T+1)(T-1) = v_j(A_i) - v_j(g_1)$. If $g_2 \in A_i$, $v_j(A_j) = N(A_j, g_1)(2T+1) + N(A_j, g_2) \leq N_(A_j, g_1)(2T+1)+T < (N_(A_j, g_1)+1)(2T+1) \leq N_(A_i, g_1)(2T+1) \leq v_j(A_i)-v_j(g_2)$. In either case, we can see that EFX is not satisfied.

By a similar argument, we can show that for any n agents and $T \neq kn$ for $k \in \mathbb{Z}^+$, $T > 1$, an EFX repeated matching need not exist. For the case that $T = kn$, matching each agent to each good k times produces an envy-free and hence EFX matching.

Failure of standard approaches to EF1 for repeated matchings. We now briefly discuss why previous algorithms for achieving EF1 do not work in the setting of repeated matchings. In the case of a round robin (Caragiannis et al. [35]) algorithm where agents get to pick a good of their choice in turn, as $n = m$, the same agents will essentially get to pick first, making the resulting matching envy-free up to T goods. Alternately reversing the order in which the agents get to pick the goods makes the matching envy-free up to $T/2$ goods.

The other well-known algorithm for finding EF1 allocations uses cycle elimination in envy graphs (Lipton et al. [82]). The algorithm allocates a good to a source in the envy-graph, and then resolves any cycles which may occur. This algorithm can't be leveraged in a matching context, as in a single round each agent must get exactly one good and even on allocating a good to the only source in the envy-graph, it may continue to remain the only source in the envy graph. We now display an algorithm that goes around this problem by finding an order to let the agents choose a good based on the envy graph.

Non-existence of EF1 repeated matchings with mixed items. We now specifically turn our attention to repeated matching instances with mixed items. Consider the following instance with $n = 2$ and $T = 1$. One of the items is a good and the other is a chore. There are exactly two possible matchings. In either, the classical extension of EF1 for mixed items from the fair division literature (e.g., see 8), which requires that the value of an agent is higher than that of another either by removing a single item from either one of the two bundles, is not satisfied. Motivated by this simple example, we propose and investigate an alternate notion of fairness to EF1 for repeated matchings, which we call *swap envy-freeness* (swapEF).

4.6 Swap Envy-Freeness

We now define a different relaxation of envy-freeness designed for such repeated one-one matchings.

Definition 4.5 (swapEF**)** *Let $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i \in \mathcal{A}}, T \rangle$ be a repeated matching instance with mixed items. A repeated matching $A = (A_1, ..., A_n)$ in I is* swapEF *if for every pair of agents $i, j \in \mathcal{A}$, either (i) or (ii) is true:*

i. $v_i(A_i) \geq v_i(A_j)$*;*

ii. There exist items $g_i \in A_i$ and $g_j \in A_j$ such that $v_i(A_i \cup \{g_j\} \setminus \{g_i\}) \geq v_i(A_j \cup \{g_i\} \setminus \{g_j\})$.

Condition (ii) requires that the value agent i has for her bundle A_i after replacing a copy of item g_i with an extra copy of item g_j is at least as high as her value for the bundle A_j of agent j after exchanging a copy of item g_j with a copy of item g_i. For instance, in the example of one good and one chore with $T = 1$, we find that while EF1 cannot be satisfied, each of the two repeated matchings are actually swapEF. This is because the agent who is matched to the chore will envy the other agent, but should the two be swapped, this envy would be mitigated. Now if $T = 2k + 1$ in this example with valuations staying constant with time, any repeated matching that matched one agent to the chore for $k + 1$ rounds will be swapEF. The agent matched to the chore for $k + 1$ rounds, will envy the other agent, but exchanging just one copy of the chore for one of the good would mitigate this envy. We first find that Algorithm 4.1 successfully finds a swapEF repeated matching, even without the non-negativity constraint on valuations (the rank definition can be trivially adapted).

4.6.1 Identical Valuations

Lemma 4.1 *Given a repeated matching instance $I = \langle \mathcal{A}, \mathcal{G}, v, T \rangle$ with identical valuations, the repeated matching returned by Algorithm 4.1 is* swapEF.

Proof: First observe that if T is an integer multiple of n, the repeated matching computed by Algorithm 4.1 creates no envy to any agent and, hence, it is swapEF as well. Now, assume that T is not an integer multiple of n; the algorithm will execute $T \bmod n$ round-robin phases in this case. Denote by $g_{i,t}$ the item agent i gets in the round-robin phase $t \in \{1, 2, ..., T \bmod n\}$. Agent i gets $\lceil T/n \rceil$ copies of each of these items, while it uses only $\lfloor T/n \rfloor$ copies of the rest. Then, for any pair of agents i and j, observe that

$$v(A_i) - v(A_j) = \sum_{t=1}^{T \bmod n} (v(g_{i,t}, \lceil T/n \rceil) - v(g_{j,t}, \lceil T/n \rceil)).$$

If $i < j$, then it is also $v(g_{i,t}, \lceil T/n \rceil) \geq v(g_{j,t}, \lceil T/n \rceil)$ for every round-robin phase t, which implies that $v(A_i) \geq v(A_j)$. The inequality is clear if both agents i and j get a copy of the same item in phase t. If this is not the case, the item agent i picks has lower rank than the item agent j picks later. This implies that $v(g_{i,t}, \lceil T/n \rceil) \geq v(g_{j,t}, \lceil T/n \rceil)$, too.

Now assume that $i > j$. By the argument above, we also get $v(A_i) \geq v(A_j)$ when agents i and j get a copy of the same item in each round. So, in the following, let us assume that this is not the case and denote by t_1 and t_2 the first and the last round-robin phase in which agents i and j get different items. Then,

$$\begin{aligned}
&v(A_i \cup \{g_{j,t_1}\} \setminus \{g_{i,t_2}\}) - v(A_j \cup \{g_{i,t_2}\} \setminus \{g_{j,t_1}\}) \\
&\quad = \sum_{t=1}^{t_1-1} (v(g_{i,t}, \lceil T/n \rceil) - v(g_{j,t}, \lceil T/n \rceil)) \\
&\quad + v(g_{j,t_1}, \lceil T/n \rceil) + \sum_{t=t_1}^{t_2-1} (v(g_{i,t}, \lceil T/n \rceil) - v(g_{j,t+1}, \lceil T/n \rceil)) \\
&\quad - v(g_{i,t_2}, \lceil T/n \rceil) + \sum_{t=t_2+1}^{T \bmod n} (v(g_{i,t}, \lceil T/n \rceil) - v(g_{j,t}, \lceil T/n \rceil)) \\
&\quad \geq v(g_{j,t_1}, \lceil T/n \rceil) - v(g_{i,t_2}, \lceil T/n \rceil) \geq 0.
\end{aligned}$$

The first inequality follows since the first and third sums are equal to 0 and the second one is non-negative. This is due to the following observations. First, notice that, by definition, both agents i and j get a copy of the same item in phases from 1 to $t_1 - 1$ and from $t_2 + 1$ to $T \bmod n$. Second, notice that the item $g_{i,t}$ that agent i picks in round-robin phase t is either the same with the one that agent j picks in the next round-robin phase $t + 1$ or one that has lower rank (and, thus, is at least as preferable). The second inequality is due to the fact that the item that agent j picks in the round-robin phase t_1 is at least as preferable to the one agent i picks later in the round-robin phase $t_2 \geq t_1$.

We have established the swapEF requirements in any case and the proof is complete. □

We now turn our attention to general valuations.

4.6.2 General Valuations

We find that we can satisfy swapEF for more general settings than EF1.

Theorem 4.6 *Given a repeated matching instance I with mixed items, n agents, and T rounds such that $T \bmod n \in \{0, 1, 2, n-2, n-1\}$, a swapEF repeated matching exists and can be computed in polynomial time.*

The proof of Theorem 4.6 uses Algorithm 4.2 from Section 4.4 for instances with $T \bmod n \in \{0, 1, 2\}$. For instances with $T \bmod n \in \{n-2, n-1\}$, we use an extension of Algorithm 4.3 from Section 4.4, which runs an additional reverse round robin phase to remove one more distinct item from each agent when $T \bmod n = n - 2$. We refer to this as Algorithm 4.4; the lines 7-12 implement the reverse round-robin phase, while the lines 1-6 are identical to Algorithm 4.3.

The properties of Algorithms 4.2 and 4.4 regarding swapEF are given by the next two lemmas, which, together with the fact that both algorithms run in polynomial time, complete the proof of Theorem 4.6.

4.6.2.1 Finding swapEF matchings when T mod n is in {0,1,2}

Lemma 4.4 *The repeated matching $A = (A_1, ..., A_n)$ produced by Algorithm 4.2 is* swapEF.

Proof: Let S denote the multiset that contains each item with multiplicity $\lfloor T/n \rfloor$. If $T \bmod n = 0$, then $A_i = S$ for every agent i and, hence, the agents are not envious of each other. If $T \bmod n = 1$, the final repeated matching is obtained after the execution of the round-robin phase. Consider two agents i and j and let g_i and g_j be the items the two agents get in this phase, respectively. If $v_i(g_i, \lceil T/n \rceil) \geq v_i(g_j, \lceil T/n \rceil)$, then

$$v_i(A_i) = v_i(S) + v_i(g_i, \lceil T/n \rceil) \geq v_i(S) + v_i(g_j, \lceil T/n \rceil) = v_i(A_j).$$

Otherwise, if $v_i(g_i, \lceil T/n \rceil) < v_i(g_j, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i \cup \{g_j\} \setminus \{g_i\}) &= v_i(S) + v_i(g_j, \lceil T/n \rceil) \\ &> v_i(S) + v_i(g_i, \lceil T/n \rceil) \\ &= v_i(A_j \cup \{g_i\} \setminus \{g_j\}). \end{aligned}$$

In both cases, the swapEF conditions are satisfied.

If $T \bmod n = 2$, the final repeated matching is obtained after the execution of the reverse round-robin phase. Consider two agents i and j. Let g_i^1 and g_j^1 be the items agents i and j get in the round-robin phase and g_i^2 and g_j^2 be the items they get in the reverse round-robin phase, respectively. We distinguish between three cases. If $|\{g_i^1, g_i^2\} \cap \{g_j^1, g_j^2\}| = 2$, then A_i and A_j are effectively identical and agent i does not envy agent j. If $|\{g_i^1, g_i^2\} \cap \{g_j^1, g_j^2\}| = 1$, assume, without loss of generality, that $g_i^1 = g_j^2 = g$ and observe that A_i has $\lceil T/n \rceil$ copies of g_i^2 and $\lfloor T/n \rfloor$ copies of g_j^1 and A_j has $\lceil T/n \rceil$ copies of g_j^1 and $\lfloor T/n \rfloor$ copies of g_i^2. If

$v_i(g_i^2, \lceil T/n \rceil) \geq v_i(g_j^1, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i) &= v_i(S \cup \{g\}) + v_i(g_i^2, \lceil T/n \rceil) \\ &\geq v_i(S \cup \{g\}) + v_i(g_j^1, \lceil T/n \rceil) = v_i(A_j). \end{aligned}$$

Otherwise, if $v_i(g_i^2, \lceil T/n \rceil) < v_i(g_j^1, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i \cup \{g_j^1\} \setminus \{g_i^2\}) &= v_i(S) + v_i(g_j^1, \lceil T/n \rceil) \\ &> v_i(S) + v_i(g_i^2, \lceil T/n \rceil) \\ &= v_i(A_j \cup \{g_i^2\} \setminus \{g_j^1\}). \end{aligned}$$

So, the swapEF conditions are satisfied.

It remains to consider the case where g_i^1, g_i^2, g_j^1, and g_j^2 are distinct. Then, A_i contains $\lceil T/n \rceil$ copies of g_i^1 and g_i^2 and $\lfloor T/n \rfloor$ copies of g_j^1 and g_j^2 and A_j contains $\lceil T/n \rceil$ copies of g_j^1 and g_j^2 and $\lfloor T/n \rfloor$ copies of g_i^1 and g_i^2. If $i < j$, agent i acts before agent j in the round-robin phase and, hence, $v_i(g_i^1, \lceil T/n \rceil) \geq v_i(g_j^1, \lceil T/n \rceil)$. If $v_i(g_i^2, \lceil T/n \rceil) \geq v_i(g_j^2, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i) &= v_i(S) + v_i(g_i^1, \lceil T/n \rceil) + v_i(g_i^2, \lceil T/n \rceil) \\ &\geq v_i(S) + v_i(g_j^1, \lceil T/n \rceil) + v_i(g_j^2, \lceil T/n \rceil) = v_i(A_j), \end{aligned}$$

and agent i does not envy agent j. Otherwise, if $v_i(g_i^2, \lceil T/n \rceil) < v_i(g_j^2, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i \cup \{g_j^2\} \setminus \{g_i^2\}) &= v_i(S) + v_i(g_i^1, \lceil T/n \rceil) + v_i(g_j^2, \lceil T/n \rceil) \\ &> v_i(S) + v_i(g_j^1, \lceil T/n \rceil) + v_i(g_i^2, \lceil T/n \rceil) \\ &= v_i(A_i \cup \{g_i^2\} \setminus \{g_j^2\}), \end{aligned}$$

and the swapEF condition is satisfied. If $i > j$, agent i acts before j in the reverse round-robin phase and, hence, $v_i(g_i^2, \lceil T/n \rceil) \geq v_i(g_j^2, \lceil T/n \rceil)$. If $v_i(g_i^1, \lceil T/n \rceil) \geq v_i(g_j^1, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i) &= v_i(S) + v_i(g_i^1, \lceil T/n \rceil) + v_i(g_i^2, \lceil T/n \rceil) \\ &\geq v_i(S) + v_i(g_j^1, \lceil T/n \rceil) + v_i(g_j^2, \lceil T/n \rceil) = v_i(A_j), \end{aligned}$$

and agent i does not envy agent j. Otherwise, if $v_i(g_i^1, \lceil T/n \rceil) < v_i(g_j^1, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i \cup \{g_j^1\} \setminus \{g_i^1\}) &= v_i(S) + v_i(g_i^2, \lceil T/n \rceil) + v_i(g_j^1, \lceil T/n \rceil) \\ &> v_i(S) + v_i(g_j^2, \lceil T/n \rceil) + v_i(g_i^1, \lceil T/n \rceil) \\ &= v_i(A_i \cup \{g_i^1\} \setminus \{g_j^1\}), \end{aligned}$$

and the swapEF condition is again satisfied. □

4.6.2.2 Finding swapEF matchings when T mod n is in {n-1,n-2}

We now show how to compute a swapEF repeated matching in the remaining cases stated in Theorem 4.6.

Algorithm 4.4: Computing a swapEF repeated matching

Input: Instance $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i \in \mathcal{A}}, T \rangle$ with $|\mathcal{A}| = n$ and $T \bmod n \in \{n-1, n-2\}$
Output: A repeated matching A

$f(i,g) \leftarrow \lceil T/n \rceil, \forall i \in \mathcal{A}, \forall g \in G$;
$P \leftarrow \mathcal{G}$ ▷ Initialize for one round of round robin ;
for $i = 1$ *to* n **do**
  $\widehat{g} \leftarrow \operatorname{argmin}_{g \in P} v_i(g, \lceil T/n \rceil)$;
  $f(i, \widehat{g}) \leftarrow f(i, \widehat{g}) - 1$;
  $P \leftarrow P \setminus \{\widehat{g}\}$;
if $T \bmod n = n - 2$ **then**
  $P \leftarrow \mathcal{G}$ ▷ Initialize for reverse order round robin ;
  for $i = n$ *to* 1 **do**
    $\widehat{g} \leftarrow \operatorname{argmin}_{g \in P} v_i(g, f(i,g))$;
    $f(i, \widehat{g}) \leftarrow f(i, \widehat{g}) - 1$;
    $P \leftarrow P \setminus \{\widehat{g}\}$;
$A \leftarrow$ GenerateFromFreq(f);

Lemma 4.5 *The repeated matching $A = (A_1, ..., A_n)$ produced by Algorithm 4.4 is* swapEF.

Proof: Let S denote the multiset that contains each item with multiplicity $\lceil T/n \rceil$. We first consider the case where $T \bmod n = n - 1$. Let i and j be two agents and denote by g_i and g_j the items that are removed from their bundles in the round-robin phase. If $v_i(g_i, \lceil T/n \rceil) \geq v_i(g_j, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i \cup \{g_i\} \setminus \{g_j\}) &= v_i(S) - v_i(g_j, \lceil T/n \rceil) \\ &\geq v_i(S) - v_i(g_i, \lceil T/n \rceil) = v_i(A_j \cup \{g_j\} \setminus \{g_i\}). \end{aligned}$$

Otherwise, if $v_i(g_i, \lceil T/n \rceil) < v_i(g_j, \lceil T/n \rceil)$, then

$$v_i(A_i) = v_i(S) - v_i(g_i, \lceil T/n \rceil) > v_i(S) - v_i(g_j, \lceil T/n \rceil) = v_i(A_j).$$

Thus, $\mathrm{swapEF}$is satisfied by Algorithm 4.4 whenever $t \bmod n = n - 1$. If $T \bmod n = n - 2$, the final repeated matching is obtained after the execution of the reverse round-robin phase. Consider two agents i and j. Let g_i^1 and g_j^1 be the items agents i and j remove in the round-robin phase and g_i^2 and g_j^2 be the items they remove in the reverse round-robin phase, respectively. We distinguish between three cases. If $|\{g_i^1, g_i^2\} \cap \{g_j^1, g_j^2\}| = 2$, then A_i and A_j are identical and agent i does not envy agent j. If $|\{g_i^1, g_i^2\} \cap \{g_j^1, g_j^2\}| = 1$, assume, without loss of generality, that $g_i^1 = g_j^2 = g$ and observe that A_i has $\lfloor T/n \rfloor$ copies of g_i^2 and $\lceil T/n \rceil$ copies of g_j^1 and A_j has $\lfloor T/n \rfloor$ copies of g_j^1 and $\lceil T/n \rceil$ copies of g_i^2. If $v_i(g_i^2, \lceil T/n \rceil) \leq v_i(g_j^1, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i) &= v_i(S \setminus \{g\}) - v_i(g_i^2, \lceil T/n \rceil) \\ &\geq v_i(S \setminus \{g\}) - v_i(g_j^1, \lceil T/n \rceil) = v_i(A_j). \end{aligned}$$

Otherwise, if $v_i(g_i^2, \lceil T/n \rceil) > v_i(g_j^1, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i \cup \{g_i^2\} \setminus \{g_j^1\}) &= v_i(S) - v_i(g, \lceil T/n \rceil) - v_i(g_j^1, \lceil T/n \rceil) \\ &> v_i(S) - v_i(g, \lceil T/n \rceil) - v_i(g_i^2, \lceil T/n \rceil) \\ &= v_i(A_j \cup \{g_j^1\} \setminus \{g_i^2\}). \end{aligned}$$

So, the swapEF conditions are satisfied in this case.

It remains to consider the case where g_i^1, g_i^2, g_j^1, and g_j^2 are distinct. Then, A_i contains $\lfloor T/n \rfloor$ copies of g_i^1 and g_i^2 and $\lceil T/n \rceil$ copies of g_j^1 and g_j^2 and A_j contains $\lfloor T/n \rfloor$ copies of g_j^1 and g_j^2 and $\lceil T/n \rceil$ copies of g_i^1 and g_i^2. If $i < j$, agent i acts before agent j in the round-robin phase and, hence, $v_i(g_i^1, \lceil T/n \rceil) \leq v_i(g_j^1, \lceil T/n \rceil)$. If $v_i(g_i^2, \lceil T/n \rceil) \leq v_i(g_j^2, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i) &= v_i(S) - v_i(g_i^1, \lceil T/n \rceil) - v_i(g_i^2, \lceil T/n \rceil) \\ &\geq v_i(S) - v_i(g_j^1, \lceil T/n \rceil) - v_i(g_j^2, \lceil T/n \rceil) = v_i(A_j), \end{aligned}$$

and agent i does not envy agent j. Otherwise, if $v_i(g_i^2, \lceil T/n \rceil) > v_i(g_j^2, \lceil T/n \rceil)$, then

$$v_i(A_i \cup \{g_i^2\} \setminus \{g_j^2\}) = v_i(S) - v_i(g_i^1, \lceil T/n \rceil) - v_i(g_j^2, \lceil T/n \rceil)$$

$$
\begin{aligned}
&> v_i(S) - v_i(g_j^1, \lceil T/n \rceil) - v_i(g_i^2, \lceil T/n \rceil) \\
&= v_i(A_i \cup \{g_j^2\} \setminus \{g_i^2\}),
\end{aligned}
$$

Hence, the swapEF condition is satisfied. If $i > j$, agent i acts before j in the reverse round-robin phase and, hence, $v_i(g_i^2, \lceil T/n \rceil) \leq v_i(g_j^2, \lceil T/n \rceil)$. If $v_i(g_i^1, \lceil T/n \rceil) \leq v_i(g_j^1, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i) &= v_i(S) - v_i(g_i^1, \lceil T/n \rceil) - v_i(g_i^2, \lceil T/n \rceil) \\ &\geq v_i(S) - v_i(g_j^1, \lceil T/n \rceil) - v_i(g_j^2, \lceil T/n \rceil) = v_i(A_j), \end{aligned}$$

and agent i does not envy agent j. Otherwise, if $v_i(g_i^1, \lceil T/n \rceil) > v_i(g_j^1, \lceil T/n \rceil)$, then

$$\begin{aligned} v_i(A_i \cup \{g_i^1\} \setminus \{g_j^1\}) &= v_i(S) - v_i(g_i^2, \lceil T/n \rceil) - v_i(g_j^1, \lceil T/n \rceil) \\ &> v_i(S) - v_i(g_j^2, \lceil T/n \rceil) - v_i(g_i^1, \lceil T/n \rceil) \\ &= v_i(A_i \cup \{g_j^1\} \setminus \{g_i^1\}), \end{aligned}$$

and the swapEF condition is again satisfied. □

Theorem 4.6 implies the following corollary.

Corollary 4.2 *In repeated matching instances with mixed items and up to five agents/items, a* swapEF *repeated matching always exists.*

We conclude this section with a comparison of EF1 and swapEF. While the two fairness notions have similar definitions, they are actually incomparable. Clearly, swapEF does not imply EF1 as it is trivially satisfied in the simple motivating example with one good and one chore presented at the beginning of this section. However, given that we use largely the same algorithms for swapEF as we did for EF1, one may believe intuitively that for goods alone, EF1 implies swapEF. This is not the case though. Consider an instance with three rounds and two agents with identical constant valuations $v(1, t) = 3$ and $v(2, t) = 2$ for two items. Giving item 1 to one agent and item 2 to the other for all three rounds is EF1 but not swapEF.

4.7 Conclusions

We introduced a new model for matchings with one-sided preferences where matchings must happen repeatedly but the valuations of agents depend on how often they have received the item in the past. We found that maximizing for social welfare here is NP-Hard in general but becomes tractable when the valuations are monotone. For fairness, we showed that more standard relaxations of envy-freeness need not exist for repeated matchings and proposed swapEF to fill this gap. An important direction of future work is resolving the existence of EF1 for goods and swapEF in general. We elaborate on these in Chapter 7.

Part 2: Two-Sided Preferences

Chapter 5

Achieving Fairness and Stability in Many-to-One Matchings

In this chapter, we investigate the unexplored area of fair and stable many-to-one matchings. Work on fairness in stable matchings with cardinal valuations is scant. We are motivated to pick leximin optimality as the notion of fairness to achieve over the space stable matchings. We first investigate matching problems with *ranked valuations* where all agents on each side have the same preference orders or rankings over the agents on the other side (but not necessarily the same valuations). For this space of problems, we provide a complete characterisation of the set of stable matchings. This leads to a novel algorithm, **FaSt**, to compute a leximin optimal stable matching under ranked *isometric* valuations. Building upon FaSt, we propose a polynomial time algorithm, **FaSt-Gen**, that finds the leximin optimal stable matching for a more general setting of ranked valuations. We next establish that, in the absence of rankings and under strict preferences, finding a leximin optimal stable matching is NP-Hard. Further, with weak rankings, the problem is strongly NP-Hard, even under isometric valuations. In fact, when additivity and non-negativity are the only assumptions, we show that, unless P=NP, no polynomial factor approximation is possible by a polynomial time algorithm.

5.1 Introduction

The various notions of fairness studied in the fair allocation literature have remained unexplored for two-sided matching settings. The exception is some very recent work [49, 59]. While these papers look at two sided fairness, they do not take into account stability. Sta-

bility is an extremely desirable property for two-sided matchings. Additionally, two-sided matching literature has largely assumed ordinal preferences. This work initiates the study of achieving fairness in stable, two-sided matchings under cardinal valuations. Motivated by certain real-world situations, we pick leximin optimality as the notion of fairness and focus on stable many-to-one matchings. We derive several algorithmic and complexity results for various special cases of many-to-one matchings. We first present a motivating example and then introduce the class of problems explored in this chapter.

Motivating Example

We consider the case of engineering college admissions in India. Here, admissions to undergraduate engineering colleges are based on a centralized examination called the Joint Entrance Exam (JEE). In 2021, over 2 million students gave the JEE [76]. Based on their performance, each student is awarded an *All India Rank* (AIR). The AIRs determine the order in which the students get to choose the college they wish to join, via a centralized admissions process. In turn, colleges are ranked by their reputation, determined by factors such as employment secured by the alumni, research output, brand, etc. For students, the reputation of a college indicates how much being an alumnus of this college will improve their future prospects.

This ranking over colleges is the same for nearly all students (except for a few who may have a strong location preference), and is influenced by rankings produced by official national rankings and reports appearing in national newspapers, magazines, media, etc. Similar to the Indian setting, central ranking procedures for college admissions are used by several countries including Brazil, Germany, Taiwan, and the UK [83].

Stability is a key criterion for college admissions. Informally, stability in this context requires that there should not exist a college-student pair where both can benefit by deviating from the prescribed matching by getting matched to each other. Baswana et. al. [20] found that using a stable matching mechanism for engineering admissions in India eliminates some of the glaring inefficiencies that were observed when stability was not considered.

Stability is not enough to guarantee fairness. Newer colleges are invariably ranked lower than well established colleges. As a result, despite some newer colleges having adequate capacity and competitive quality, it happens that they are ignored by students, particularly the top ranking students. Matchings where the large majority of students are matched to only well established (higher ranked) colleges may be stable but are clearly unfair to newer colleges. This results in poor reputation and a loss of opportunity for the newer colleges. Consequently, these colleges continue to be poorly ranked in future years as well. Hence, in

the absence of any explicit intervention to ensure some degree of fairness, this exacerbates the discrepancy among the colleges for future batches of students.

In essence, this could create a vicious, self-reinforcing hierarchy. To offset this discrepancy, a suitable notion of fairness could be implemented in the centralized admission process. Should the worthy newer colleges get more students, which they deserve, they would have an opportunity to excel and improve their rankings. Naturally, in our attempt to help these colleges, we must not ignore the interests of the students. That is, in order to help newer colleges, we cannot send top-ranked students to low-ranked colleges. Fairness must be maintained for the students as well as the colleges.

Ranked Valuations

Several practical many-to-one matching settings belong to the *ranked valuation* setting where there are inherent rankings across the agents on each side. The class of ranked valuations has been well studied in the context of fair division as well as stable matchings with ordinal preferences (with a variety of nomenclature) [4, 10, 30, 69]. This class is a generalization of identical valuations, a very well studied class of valuations [11, 16, 41, 96]. In ranked valuations, for each side, all agents have the same preference orders or rankings over the agents on the other side (but may not have the same valuations, which would be required for identical valuations). For example, in labour market settings, workers are ranked based on their experience, while employers may be ranked on the wages they offer. In the college admissions example being discussed, the ranking of colleges is the same for nearly all students and the colleges have a single ranking over the students (based on AIRs) (except for a few who have a strong location preference). This ranking is usually influenced by ranking surveys and word-of-mouth communication. These rankings lead to a structure over the space of stable matchings.

Isometric Valuations

We also investigate the isometric valuations setting, where the valuation of a student for a college is the same as that of the college for the student. This assumption captures several practical settings. It is generally observed in India (and this may be true for many other countries as well), that students choose to pursue technical degrees, in large part, with the expectation of a well paying job upon graduation. The jobs that the students secure, in turn, determine the reputation of their colleges, especially for newer colleges. Consequently, in this setting, the value that a student and a college gain from each other is the expected quality of future opportunities awaiting the student. Further, the compensation packages students receive are also influenced by the reputation of the colleges they have attended. As a result,

the rankings play an important role. Finding fair and stable matchings is practically relevant with isometric valuations as well.

Leximin Optimality

While there is a wide variety of fairness notions prevalent in prior work, given that they focus on allocation settings, the large majority of these need not always maintain fairness across the two sets of agents. The question of whether agents on both sides must be treated equally is up for debate. In this work, we assume that agents from either side are equally important. To this end, we focus on *leximin optimality* [23, 45, 46, 96] as our notion of fairness. Informally, a leximin optimal matching is one that maximizes the value of the worst-off agent (i.e. satisfies the maximin/santa claus criteria), and out of the matchings that achieve this, maximizes the value of the second worst-off agent, and so on. Essentially, this minimizes the discrepancy in the values achieved by each of the agents on either side, ensuring a fair balance in the interests of both sides. For college admissions, leximin optimality improves values achieved by lower ranked students and colleges with minimal compromise in the matchings of higher ranked colleges and students.

An attractive aspect of the leximin notion is that it is an optimization based fairness notion with a guaranteed optimal outcome over stable matchings. While notions like egalitarian welfare (also called maximin welfare or the Santa Claus problem) and Nash social welfare are also guaranteed to exist, they are not sufficient for ensuring fairness for agents on both sides. Other popular fairness notions like envy-freeness and equitability need not coexist with stability. We defer a discussion on how these fairness notions interact with stability and why they do not extend well to the case of two-sided preferences to Appendix B.1 As a result, leximin turns out to be an appealing notion of fairness for two-sided matchings, worthy of further investigation.

5.1.1 Technical Contribution of this Chapter

Our paper in 2020 [91] was the first attempt to study fairness and stability with cardinal valuations. In this work, we seek to find a many-to-one matching that is a leximin optimal over the space of stable matchings. Table 5.1 summarises our contributions. Here, m and n are the number of colleges and students, respectively. Recall that strict preferences mean that no agent is indifferent about any agent on the other side (each agent's valuations imply a strict linear order over agents on the other side). Weak preferences allow agents to be indifferent between two agents (their valuations imply a weak order on the agents on the other side).

Nature of Valuations	Strict Preferences	Weak Preferences
Ranked + Isometric	$O(mn)$ (Thm 5.1)	Strongly NP-Hard (Thm 5.5)
Ranked Valuations	$O(m^2n^2)$ (Thm 5.2)	APX-Hard (Thm 5.6)
General with $m = 2$	$O(n^2)$ (Thm 5.3)	NP-Hard (Thm 5.5)
General	NP-Hard (Thm 5.4)	APX-Hard (Thm 5.6)

Table 5.1: Summary of Results

We find that when there are strict rankings over either side, the space of stable matchings has a structure. Moreover, we can find a leximin optimal stable matching in polynomial time. In particular, for rankings with isometric valuations (settings where the valuation of a student for a college is equal to the college's valuation for the student), we present an algorithm, which we call FaSt (Fair and Stable) that outputs a leximin optimal stable matching in $O(mn)$ time.

We then extend this algorithm to present an $O(m^2n^2)$ time algorithm for general ranked valuations. In the absence of rankings, the structure implied by rankings is no longer present, and as a result, there is no tractable way to iterate over the space of stable matchings. However, when there are exactly two colleges, we can in fact iterate over the space of stable matchings when preferences are strict. Using this property, we present an algorithm which runs in time $O(n^2)$. Our algorithms all follow a similar pattern, which only requires a certain way of iterating over the space of stable matchings. We believe this is a novel approach for optimizing over the space of matchings.

Unfortunately, these ideas do not extend to more general settings. In Theorem 5.4, we show that in the absence of rankings, even with strict preferences, finding the leximin optimal stable matching is intractable, when the number of colleges in unconstrained. We further establish, in Theorem 5.5, that it is also NP-Hard to find a leximin optimal stable matching even under isometric valuations and with weak rankings and $m = 2$. In fact, when the valuation functions are unconstrained beyond being additive, we find, in Theorem 5.6, that it is NP-Hard to obtain a polynomial factor approximation on the leximin optimal stable matching, even with weak rankings.

5.2 Notation and Preliminaries

In this section, we first set up our model, provide the necessary definitions, discuss the relevant work, and state the main results.

5.2.1 Definitions and Notation

Valuations

Let $\mathcal{S} = \{s_1, \ldots, s_n\}$ and $\mathcal{C} = \{c_1, \ldots, c_m\}$ be non-empty, finite, and ordered sets of students and colleges, respectively. We assume that there are at least as many students as colleges, that is, $n \geq m$. We shall assume that each college has a capacity (or budget) $b_j \leq n$ on the maximum number of students that can be matched to it. Further, we assume that $\sum_j b_j \geq n$, that is, each student will always be matched to a college.

Let $u_i(\cdot)$ and $v_j(\cdot)$ be the valuation functions of student s_i, and college c_j, $i \in [n]$, $j \in [m]$. This chapter studies only non-negative and additive valuation functions. We define $U = (u_1, \cdots, u_n)$ and $V = (v_1, \cdots, v_m)$ and $B = (b_1, \cdots, b_m)$. Hence, an instance of *stable many-to-one matchings* (SMO) is captured by the tuple $I = \langle \mathcal{S}, \mathcal{C}, U, V, B \rangle$. The ordering implied by an agent's valuations over agents from the other set are called preferences. Strict preferences require that the preference order is strict, or in other words, there for any agent, there are no two agents in the other set for whom they have equal value. Weak preferences imply that there may be ties in the preference order.

We shall also look at *isometric valuations*[1] where the value that a student s_i has for a college c_j is the same as c_j's value for s_i, that is, $u_i(c_j) = v_j(s_i)$, for all $i \in [n]$ and all $j \in [m]$. Each instance with isometric valuations can be captured by the tuple $I = \langle \mathcal{S}, \mathcal{C}, V, B \rangle$. Here V can equivalently represent an $n \times m$ matrix where V_{ij} is the valuation of s_i for matching with c_j, or $V_{ij} = u_i(c_j) = v_j(s_i)$.

Rankings or *ranked valuations* imply that for all students, while the exact values need not be the same, the (strict) ordering over colleges is the same and analogously, each college has the same preference order over students. To this end, for convenience we assume that whenever the valuations are ranked, $u_i(c_1) > u_i(c_2) > \cdots > u_i(c_m)$ for all $i \in [n]$ and $v_j(s_1) > v_j(s_2) > \cdots > v_j(s_n)$ for all $j \in [m]$. Weak rankings shall allow for ties, i.e., $u_i(c_1) \geq u_i(c_2) \geq \cdots \geq u_i(c_m)$ for all $i \in [n]$ and $v_j(s_1) \geq v_j(s_2) \geq \cdots \geq v_j(s_n)$ for all $j \in [m]$. Thus, for ranked isometric valuations, each row and column of V are sorted in decreasing order.

[1] We avoid the term symmetric valuations as it has previously been used in two-sided and one-sided matching literature with different meanings.

Our goal is to find a many-to-one matching μ of the bipartite graph $G = (\mathcal{S}, \mathcal{C}, \mathcal{S} \times \mathcal{C})$ such that μ satisfies stability as well as fairness properties. A matching $\mu \subseteq \mathcal{S} \times \mathcal{C}$ is a subset of $\mathcal{S} \times \mathcal{C}$ such that each student has at most one incident edge present in the matching, and the number of incident edges on a college is *at most* their capacity b_j, i.e., for any $i \in [n]$ and $j \in [m]$, $|\mu(s_i)| \leq 1$ and $|\mu(c_j)| \leq b_j$.

The valuation of a student s_i under a matching μ is written as $u_i(\mu) = u_i(\mu(s_i))$. For a college c_j the valuation under matching μ is defined as $v_j(\mu) = \sum_{s_i \in \mu(c_j)} v_j(s_i) \geq 0$.

For the majority of this chapter, we shall assume that $b_j = n$. That is, each college can accommodate as many students as needed. Consequently, we shall study instances of the type $\langle \mathcal{S}, \mathcal{C}, U, V \rangle$. This assumption does not affect the hardness results as under budgets, we can simply assume that $b_j = n$ for all j. For the algorithms, the budgeted setting would require extra work. However, we find that even with budgets, the reasoning behind our algorithms can be extended, with some more intricate checks required. The algorithms for the budgeted setting are presented in Appendix B.3.

Fairness and Efficiency

Next, we define our two central properties for many-to-one matchings, namely, stability and leximin optimality.

Definition 5.1 (Stable Matching) *A matching μ of instance $I = \langle \mathcal{S}, \mathcal{C}, U, V \rangle$ is said to be stable if no (s_i, c_j) is a* blocking pair *for μ.*

Definition 5.2 (Blocking Pair) *Given a matching μ, (s_i, c_j) form a* blocking pair *if $s_i \notin \mu(c_j)$ and there exists $s_{i'} \in \mu(c_j)$, $c_{j'} = \mu(s_i)$ such that $v_j(s_i) > v_j(s_{i'})$ and $u_i(c_j) > u_i(c_{j'})$. That is s_i and c_j prefer each other to (one of) their partners under μ.*

Note that this definition of stability has also been called pairwise stability and justified envy-freeness in prior work. Stability often requires a non-wastefulness constraint as well. However, prior work [123, 126] has found that non-wastefulness and pairwise stability are incompatible under distributional constraints, aimed at ensuring fairness. Thus, we do not demand our stable matchings to be non-wasteful. We denote the space of complete stable matchings, matchings which are stable and do not leave any agent unmatched, as $\mathcal{S}_C(I)$.

Our work aims to find the leximin optimal over the space of stable matchings. The most convenient way of identifying such a matching is using leximin tuples. The leximin tuple of any matching is simply the tuple containing the valuations of all the agents (students and colleges) under this matching, listed in non-decreasing order. Hence, the position of an

agent's valuation in the leximin tuple may change under different matchings. The leximin tuple of a matching μ will be denoted by $\mathcal{L}_\mu$. The t^{th} index of $\mathcal{L}_\mu$ is denoted by $\mathcal{L}_\mu[t]$.

Definition 5.3 (Leximin Domination) *We say that matching μ_1 leximin dominates μ_2 if there exists a valid index k such that $\mathcal{L}_{\mu_1}[k'] = \mathcal{L}_{\mu_2}[k']$ for all $k' < k$ and $\mathcal{L}_{\mu_1}[k] > \mathcal{L}_{\mu_2}[k]$.*

We shall say that the leximin value of μ_1 is greater than that of μ_2 if μ_1 leximin dominates μ_2. This shall be denoted by $\mathcal{L}_{\mu_1} > \mathcal{L}_{\mu_2}$. We shall say that μ_2 is leximin inferior to μ_1, when μ_1 leximin dominates μ_2.

Definition 5.4 (Leximin Optimal) *A leximin optimal matching μ^* is one that is not leximin dominated by another matching.*

Essentially, we wish to find a matching that maximizes the left most value in the leximin tuple, of those that do, find the one that maximizes the second value and so on. This is NP-Hard, in general [23]. Our goal is to find a *leximin optimal over all stable matchings*.

5.2.2 Overview of Main Results

The goal of this chapter is to find a stable and leximin optimal many-to-one matching. Leximin is intractable in general, so we first look at ranked valuations, where the space of stable matchings has an appealing structure that we can exploit.

Lemma 5.1 *Given an instance of ranked valuations, a matching μ is stable if and only if, for all colleges $j \in [m]$, $\mu(c_j) = \{s_{w_j+1}, \cdots, s_{w_j+k_j}\}$ where $k_j = |\mu(c_j)|$ and $w_j = \sum_{t=1}^{j-1} k_t$.*

Lemma 5.1 ensures that a stable solution for a ranked instance would necessarily match a contiguous set of students to each c_j. We exploit this in the algorithm FaSt (Algorithm 5.2), which runs in time $O(mn)$. Its correctness is established in the following theorem.

Theorem 5.1 *FaSt (Algorithm 5.2) finds a leximin optimal stable matching for ranked isometric valuations in time $O(mn)$.*

Building on this algorithm, we develop another algorithm for general ranked valuations FaSt-Gen (Algorithm 5.3) which runs in time $O(m^2n^2)$.

Theorem 5.2 *FaSt-Gen (Algorithm 5.3) finds a leximin optimal stable matching given an instance of general ranked valuations in time $O(m^2n^2)$.*

When there are only two colleges and strict preferences (but no global rankings over colleges and students), we show that there exists a polynomial time algorithm to find the leximin optimal stable matching in this setting.

Theorem 5.3 *FaSt-Const (Algorithm 5.5) finds a leximin optimal matching given an instance with strict preferences and $m = 2$ in time $O(n^2)$.*

Unfortunately, we cannot expect tractable algorithms for much more general settings. In Section 5.4, we find that under strict preferences, without an assumption of rankings, the problem of finding a leximin optimal stable matching is NP-Hard when $n = \Omega(m)$.

Theorem 5.4 *It is NP-Hard to find a leximin optimal stable matching under strict preferences.*

The hardness of finding the leximin optimal stable matching doesn't follow from the hardness of leximin in the allocations, as stable matchings under strict preferences can't capture all feasible allocations. Given that rankings in combination with strict preferences give rise to polynomial time algorithms, a natural question is whether rankings with weak preferences can lead to polynomial time algorithms. This is not the case. We show that even on relaxing strict rankings to weak rankings, finding the leximin optimal stable matching under isometric valuations is intractable, even with a constant number of colleges.

Theorem 5.5 *It is NP-Hard to find the leximin optimal stable matching under isometric valuations with weak rankings with $m = 2$ and strongly NP-Hard with $n = 3m$.*

Clearly, this implies that it is NP-Hard to find the leximin optimal stable matching in general. However, in the absence of isometric valuations, the problem is harder still. For the general problem, we establish APX-Hardness.

Theorem 5.6 *Unless P=NP, for any $\delta > 0$, $c \in \mathbb{Z}^+$ there is no $1/cn^{\delta}$-approximation algorithm to find a leximin optimal stable matching under unconstrained additive valuations.*

Before we discuss these impossibility results, we detail our algorithmic results, starting with the setting of ranked isometric valuations.

5.3 Algorithmic Results

Most of the algorithms in this chapter follow a similar structure: initialize with the student optimal stable matching and then iterate over the space of stable matchings to look for leximin value improvements. Doing this in polynomial time with a termination guarantee requires a method that makes small changes with positive leximin improvements at every iteration and ensures stability as an invariant in each iteration.

Here, our algorithms rely on strict preferences and ranked valuations. Recall that for any $i \in [n]$ and $j \in [m]$, $u_i(c_1) > \cdots > u_i(c_m)$ and $v_j(s_1) > \cdots > v_j(s_n)$. We first show how these properties provide structure over the space of stable matchings. We find that for a matching to be stable, it must be in accordance with the rankings.

Lemma 5.1 *Given an instance of ranked valuations, a matching μ is stable if and only if, for all colleges $j \in [m]$, $\mu(c_j) = \{s_{w_j+1}, \cdots, s_{w_j+k_j}\}$ where $k_j = |\mu(c_j)|$ and $w_j = \sum_{t=1}^{j-1} k_t$.*

Proof: We prove the forward implication by assuming μ to be a stable matching. Let μ match each college c_j to k_j students. We inductively prove that the required property holds. We first show that $\mu(c_1)$ is the set of first k_1 students that is, $s_1, \cdots, s_{k_1}$. If $k_1 = 0$, this is trivially satisfied. We can thus assume that $k_1 > 0$. Suppose c_1 is not matched to all of $s_1, \cdots, s_{k_1}$ under μ. As a result, there exists some $i \in [k_1]$ such that $s_i \notin \mu(c_1)$ and that $\mu(s_i) = c_j$ for some $j > 1$. Consequently, there must be an $i' > k_1$ such that $s_{i'} \in \mu(c_1)$. However, $u_i(c_1) > u_i(c_j)$ and $v_1(s_i) > v_1(s_{i'})$, by assumption. Thus, (s_i, c_1) form a blocking pair, which contradicts the fact that μ is a stable matching.

We now assume that, for the stable matching μ, the lemma holds for the first $t-1$ colleges, $t \geq 2$ i.e., $\mu(c_j) = \{s_{w_j+1}, \cdots, s_{w_j+k_j}\}$ where $w_j = \sum_{j'=1}^{j-1} k_{j'}$ for all $j < t$.

We now show that the lemma is true for c_t. Suppose not, then there exists i such that $w_t < i \leq w_t + k_t$ and $s_i \notin \mu(c_t)$. This implies that, there must exist some $i' > w_t + k_t$ such that $s_{i'} \in \mu(c_t)$. Let $\mu(s_i) = c_j, j > t$. Now, as the valuations are ranked, $u_i(c_t) > u_i(c_j)$ and $v_t(s_i) > v_t(s_{i'})$. This implies that (s_i, c_t) form a blocking pair which contradicts the stability of μ. This proves that if μ is a stable matching that matches c_j to k_j students , then $\mu(c_j)$ must be equal to $\{s_{w_j+1}, \cdots, s_{w_j+k_j}\}$.

We now prove the reverse implication. Let μ be a matching which matches c_j to k_j students and $\mu(c_j) = \{s_{w_j+1}, \cdots, s_{w_j+k_j}\}$ where $w_j = \sum_{j'=1}^{j-1} k_{j'}$ for all $j \in [m]$.

Fix $i \in [n]$, and let $\mu(s_i) = c_j$. If $j = 1$, s_i clearly does not form a blocking pair with any college as they are matched to their most preferred college. If $j > 1$, then A_i prefers $c_1, \cdots, c_{j-1}$ to c_j. But, all these colleges prefer each of the students matched to them to s_i,

because of the ranking. Consequently, s_i does not form any blocking pairs. As a result, there are no blocking pairs in μ and it is a stable matching. □

Note that the number of *complete* stable matchings under ranked instances is $\binom{n+1}{m-1}$, hence, brute force would not be tractable unless the number of colleges is constant. We first look at a subset of matching instances, called *ranked isometric valuations* as a stepping stone to a more general result.

Lemma 5.1 in conjunction with isometric valuations, provides a structure over the leximin values of stable matchings, under ranked valuations. This enables us to find the leximin optimal stable matching. We now list some observations about the structure of a leximin optimal stable matching μ^* over the set of stable matchings under ranked valuations.

Observation 5.1 *No agent is unmatched under a leximin optimal stable matching μ^*, that is, $\mu^*(a) \neq \emptyset$ for any $a \in \mathcal{S} \times \mathcal{C}$.*

This in conjunctions with Lemma 5.1 leads to the following observation.

Observation 5.2 *Under ranked valuations, in the leximin optimal stable matching μ^*, $s_1 \in \mu^*(c_1)$ and $s_n \in \mu^*(c_m)$.*

For other students, we are only able to guarantee the following as a consequence of Lemma 5.1.

Observation 5.3 *If s_i is matched to c_j then s_{i-1} must be matched to either c_j or c_{j-1}. That is, $s_i \in \mu^*(c_j) \;\Rightarrow\; s_{i-1} \in \Big(\mu^*(c_j) \cup \mu^*(c_{j-1})\Big)$.*

We now use these observations, first in the specific context of ranked isometric valuations.

5.3.1 Leximin for Ranked Isometric Valuations

For ranked isometric valuations we have two additional observations which are instrumental in the execution of our algorithm. They rely on the fact that due to isometric valuations, if $1 \leq i < i' < n$ for any $j \in [m]$, $v_j(s_i) > v_j(s_{i'})$, thus, $v_i(c_j) > v_{i'}(c_j)$.

Observation 5.4 *Under ranked isometric valuations, for each complete stable matching $\mu \in \mathcal{S}_C(I)$, the values of the students will appear in order of their rank in the leximin tuple, that is, for any $i < i'$, $u_{i'}(\mu) < u_i(\mu)$.*

Observation 5.5 *For any matching $\mu \in \mathcal{S}_C(I)$, under isometric valuations, $v_j(\mu) \geq u_i(\mu)$ for each college c_j and for all $s_i \in \mu(c_j)$.*

These observations are critical for the particular design of FaSt (Algorithm 5.2), which outputs a leximin optimal stable matching under ranked isometric valuations in time that is linear in the size of the input.

FaSt: An Algorithm to Find a Fair and Stable Matching

We first present an $O(mn)$ time algorithm, called FaSt, to find a leximin fair stable matching under ranked isometric valuations. Recall that for isometric valuations, V also denotes an $n \times m$ matrix where $V_{ij} = u_i(c_j) = v_j(s_i)$. For ease of presentation, we shall assume that that there are no capacity constraints, that is, for any college c_j, $b_j = n - 1$, that is we effectively assume no capacity constraints. However, this is not a particularly limiting assumption. The full algorithm with unrestricted b_js is given in Appendix B.3 and follows the same logic. The time complexity of both the versions of the algorithm is $O(mn)$. We now use the structure outlined in the previous subsection to give an $O(mn)$ time algorithm for finding the leximin optimal stable matching.

In essence, the algorithm starts with the student optimal complete stable matching and gradually finds leximin optimal matching by improving the valuations of the colleges in increasing order of rank, keeping stability and non-zero valuations for all agents as an invariant. By Observation 5.3 and 5.5, we can start with student s_n and iteratively decide the matchings for higher ranked students .

The initial stable solution, which is the student optimal stable matching, matches student s_n to college c_m and student s_1 to college c_1 (using Observation 5.2). Further, the first $n - (m-1)$ students are matched to c_1. For the remaining colleges get only one student each, $\mu(c_j) = s_{n-(m-j)}$ for each $j \geq 2$. The algorithm then systematically increases the number of students matched to the lowest ranked college c_m till there is a decrease in the leximin value, at which point, we switch to the next lowest ranked college and repeat. During this, the number of students matched to all other colleges except c_1 remains fixed. We call this procedure 'Demote' algorithm (Algorithm 5.1).

We shall say that the matching of a student s_i (or a college c_j) is *fixed* if it will not change any further during the execution of FaSt. The algorithm stops when one of the two happens: either the bottom $m-1$ colleges : $c_2, \cdots, c_m$ get fixed, or the c_1 is matched to s_1 only. Note that, every time a student is sent to a lower ranked college or demoted, no student sees an increase in their valuation and no college, other than c_1, sees a decrease in their valuation. Hence, whenever a student is demoted, their position in the leximin tuple either moves to the left or stays in the same position, and the position of the corresponding college moves to the right. Due to Observation 5.4, the valuations of the previously matched students are

unaffected. Hence, the algorithm optimizes for one leximin position at a time, before moving on to the next.

Demote Overview. Demote (Algorithm 5.1) maintains the invariant of a complete stable matching. If we decide to send student s_i to college c_j when they are currently matched to c_{j-1}, it is not enough to simply do this one step. Doing this alone will make c_{j-1} unmatched, which violates our invariant. As a result, we need to match the least preferred student matched $\mu(c_{j-2})$ to c_{j-1}. For isometric valuations, this is s_{i-1}. This must continue till we send c_1's lowest ranked matched student to c_2. For this to be feasible, c_1 must be matched to at least 2 students. If not, no transfers are possible and as a result, no further improvement can be made to the leximin tuple. We ensure this feasibility in the `while` loop condition in Step 7 of FaSt.

Algorithm 5.1: Demote

Input: A matching μ, college indices $down$ and up.
Output: μ

Set $p \leftarrow down$;
while $p > up$ **do**
  $t \leftarrow \text{argmin}_{i:s_i \in \mu(c_{p-1})} v_{p-1}(s_i)$;
  $\mu(c_{p-1}) \leftarrow \mu(c_{p-1}) \backslash \{s_t\}$;
  $\mu(c_p) \leftarrow \mu(c_p) \cup \{s_t\}$;
  $p \leftarrow p - 1$ ▷ Move s_i to c_{down} while maintaining the number of students matched to all other colleges except c_{up}. ;

Theorem 5.1 *FaSt (Algorithm 5.2) finds a leximin optimal stable matching for ranked isometric valuations in time* $O(mn)$.

Proof: We first prove the correctness of FaSt.
Correctness: During initialization, FaSt matches s_n to c_m, and fixes the match by assigning s_n to the set $\mathcal{F}$. This step indicates that s_n remains matched to c_m throughout the execution of the algorithm. Recall that, by Observation 5.4, for the leximin optimal stable matching μ^*, the first entry in its leximin representation $\mathcal{L}(\mu^*)$ is $V_{nm} = u_n(c_m)$, which is ensured by FaSt.

We shall say that the matching of a student s_i (or a college c_j) is *fixed correctly* if the it is matched as in a leximin optimal matching. Let the matchings of $s_{t+1}, \cdots, s_n$ be fixed, $t \leq n-1$ and let s_{t+1} be matched to c_d. Hence, the matchings of $c_{d+1}, \cdots, c_m$ are also fixed correctly.

By Observation 5.4, $c_1, \cdots, c_d$ and $s_1, \cdots, s_{t-1}$ must occur to the right of s_t in the leximin tuple of the leximin optimal stable matching. Similarly, $s_{t+1}, \cdots, s_n$ must be listed to the left

Algorithm 5.2: FaSt

Input: Instance of ranked isometric valuations $\langle \mathcal{S}, \mathcal{C}, V \rangle$
Output: μ

Initiate a stable matching: $\mu(c_1) \leftarrow \{s_1, \cdots, s_{n-m+1}\}$ and $\mu(c_j) \leftarrow \{s_{n-(m-j)}\}$ for $j \geq 2$;
Initialize $i \leftarrow n-1$, $j \leftarrow m$;
Set $\mathcal{L}$ as the leximin tuple for μ;
Set $pos[i]$ as the position of s_i in $\mathcal{L}$, $i \in [n]$;
▷ tie breaking for position in $\mathcal{L}$: agents who attain the same value are listed in increasing order of rank, with student before colleges .
Initialize $\mathcal{F} \leftarrow \{s_n\}$ ▷ stores the agents whose matching is fixed;
while $i > j-1$ *AND* $j > 1$ **do**
  if $v_j(\mu) \geq V_{i(j-1)}$ **then**
    $j \leftarrow j-1$;
  else
    if $[V_{ij} > v_j(\mu)]$ **then**
      $\mu \leftarrow Demote(\mu, j, 1)$;
    else
      if $V_{ij} < v_j(\mu)$ **then**
        $j \leftarrow j-1$;
      else
        $k \leftarrow i$, $t \leftarrow pos[i]$, and
        $\mu' \leftarrow \mu$ ▷ Look ahead: does sending s_i to c_j improve leximin;
        while $k > j-1$ **do**
          if $V_{kj} > \mathcal{L}[t]$ **then**
            $i \leftarrow k$ and $\mu \leftarrow Demote(\mu', j, 1)$;
            break;
          else
            if $V_{ij} < v_j(\mu)$ **then**
              $j \leftarrow j-1$;
              break;
            else
              $\mu' \leftarrow Demote(\mu', j, 1)$ ▷ Another tie, send s_k to c_j tentatively;
              $k \leftarrow k-1$, and $t \leftarrow t+1$;
        if $k = j-1$ *AND* $\mu \neq \mu'$ **then**
          $j \leftarrow j-1$;
  $\mathcal{F} \leftarrow \{s_i, \cdots, s_n\} \cup \{c_{j+1}, \cdots c_m\}$;
  $Update(\mathcal{L}, \mu, pos)$;
  $i \leftarrow i-1$;

of s_t. This will not change, irrespective of the way the other agents are matched, as long as stability is maintained. Also, since s_t's valuation for c_d does not depend on how any other

agent is matched, it will not change once it is fixed. As a result, μ^* matches s_t to c_d if and only if it results in a leximin dominating matching.

By sending s_t to c_d, s_t moves to the left of its current position in the leximin tuple. To improvement the leximin value, its value at the new position should not be lower than the current value there. Thus, if V_{td} is less than c_d's current valuation, s_t must be fixed to c_{d-1}, the matching of c_d must be matched as in the current matching. Even if we add more students to c_d's matching, s_t will remain at the same position (due to Lemma 5.1) and as a result will be leximin inferior to the current matching.

In the case where these two values are equal, we must look ahead to see whether by demoting s_t we eventually result in an increased leximin value. Simply comparing c_d's new valuation need not suffice. By demoting s_t we may land in a setting where c_d's new valuation is less than $V_{t(d-1)}$, resulting in a matching which is leximin inferior to the current. However by sending sending more students to c_d, we may improve upon the current leximin tuple. Let the position of s_t when matched to c_d be k. Thus, we must compare the new leximin values (obtained by sending s_{t+1} to c_d) at position $k+1$ with the current leximin value at position $k+1$. If the values are equal we may have to look ahead further and accordingly fix the matching. Thus, when the matchings of s_{t+1} and s_n are fixed we can correctly fix s_t as in a leximin optimal matching.

Termination: The algorithm considers each student - college pair at most once. All computations during one such considerations can be done in constant time. The update to the leximin tuple $\mathcal{L}$ and the array pos can be performed in time $O(n+m) = O(n)$ as well. Thus, the time taken is $O(mn)$ in the worst case, which is linear in the number of edges of the underlying bipartite graph. □

The success of this approach is contingent on i) the inherent rankings, ii) valuation functions of the colleges being additive, and iii) the five observations listed earlier. We do not rely directly on the fact that the valuations are isometric. Consequently, whenever we have these three requirements met in a matchings instance, we can use FaSt as is to find the leximin optimal stable matching. Essentially, FaSt works correctly for the space of ranked instances where the valuations are additive and follow the following restriction: $u_i(c_j) \leq v_j(s_i)$ and $u_1(c_j) \geq u_2(c_j) \geq \cdots \geq u_n(c_j)$ for all $i \in [n]$ and $j \in [m]$. The extension of FaSt for instances without the assumption that $b_j = n-1$ is detailed in Appendix B.3.

5.3.2 General Ranked Valuations

We now discuss the algorithm to find the leximin optimal stable matching under general ranked valuations where values across an edge of the bipartite graph need not be the same. For ease of presentation, we shall assume that the capacity of each c_j, $b_j = n - 1$. As in the case of isometric valuations, this is not a particularly simplifying assumption. The complete algorithm without this assumption is deferred to Appendix B.3. The time complexity and the idea behind the two algorithms remains the same.

We no longer assume any relation between the u_is and v_js. Consequently, Observations 5.4 and 5.5 stated in the start of the section, need not hold any longer. Hence, for the general ranked setting, the approach followed in Algorithm 5.2 will no longer be applicable to find a leximin optimal stable matching, beyond the case when $m = 2$.

The approach behind FaSt, essentially starts with the student optimal stable matching and increases the number of students matched to the lowest ranked college till there is a decrease in the leximin value. Then it moves to the next lowest ranked college and repeats. Without Observation 5.5, we may have that the colleges lie to the left of the students in the leximin tuple. As a result, we may continue to increase the valuation of the lowest ranked college without much increase in the valuations of other colleges which may now lie to the left of c_m. Consequently, we may end up only balancing the values of c_1 and c_m, not finding the leximin optimal.

FaSt-Gen

For the general ranked valuations setting, we propose an approach that builds on FaSt. It also starts with the student optimal stable matching. In each iteration, we increase the number of students matched to the leftmost unfixed college by one if it increases the leximin value. If not, the algorithm fixes the upper limit of this college and the lower limit of the next highest ranked one. In order to do this, we decrease one student from a higher ranked college using the demote procedure (Algorithm 5.1). The choice of this higher ranked college is c_1 initially, till it can no longer give out any more students, then we consider the next college whose lower limit is not fixed. Thus, the algorithm gradually fixes the upper and lower limits of the students matched to each college.

There is one more subtle feature to note. In the absence, of Observation 5.4, when increasing the number of students of a particular college a leximin decrease may happen due to a higher ranked student-college pair. That is, a student which is being moved, but not to the lowest unfixed college may cause a leximin decrease. In such cases, we temporarily fix or "soft fix" the upper limits of some colleges and then unfix them, once the college which

caused the leximin decrease starts giving out students currently matched to it. We assume a routine we call $sourceDec(\mu_1, \mu_2)$ which returns the agent who is the cause of the leximin decrease in μ_1 vs μ_2.

Algorithm 5.3: FaSt-Gen

Input: Instance of general ranked valuations $\langle \mathcal{S}, \mathcal{C}, U, V \rangle$
Output: μ

$\mu(c_1) \leftarrow \{s_1, \cdots, s_{n-m+1}\}$ and $\mu(c_j) \leftarrow \{s_{n-(m-j)}\}$ for $j \geq 2$ ▷ Initialize a stable matching;
$UpperFix \leftarrow \{c_1\},\ LowerFix \leftarrow \{c_m\}$;
$SoftFix \leftarrow \emptyset$;
$Unfixed \leftarrow UpperFix^c$;
while $|UpperFix \backslash LowerFix| + |LowerFix| < m$ **do**
  $up \leftarrow \min_{j \notin LowerFix} j$;
  $down \leftarrow \text{argmin}_{j \in Unfixed}\, v_j(\mu)$.;
  $SoftFix \leftarrow SoftFix \backslash \{(j, j') | j' \leq up < j\}$;
  if $|\mu(c_{up})| = 1$ *OR* $v_{up}(\mu) \leq v_{down}(\mu)$ **then**
    $LowerFix \leftarrow LowerFix \cup \{c_{up}\}$;
  else
    $\mu' \leftarrow Demote(\mu, down, up)$;
    if $\mathcal{L}_{\mu'} \geq \mathcal{L}_{\mu}$ **then**
      $\mu \leftarrow \mu'$;
    else
      ▷ Decrease in leximin value, need to check the source of the decrease;
      if $sourceDec(\mu', \mu) = c_{up}$ **then**
        $LowerFix \leftarrow LowerFix \cup \{c_{up}\}$;
        $UpperFix \leftarrow UpperFix \cup \{c_{up+1}\}$
      else
        if $sourceDec(\mu', \mu) \in \mathcal{S}$ **then**
          $c_t \leftarrow \mu(sourceDec(\mu', \mu))$;
          $LowerFix \leftarrow LowerFix \cup \{c_t\}$;
          $UpperFix \leftarrow UpperFix \cup \{c_{t+1}\}$;
          $A \leftarrow \{j | j > t+1, j \in Unfixed\}$;
          $SoftFix \leftarrow SoftFix \cup (A \times \{t+1\})$;
        else
          $(\mu, LowerFix, UpperFix, SoftFix) \leftarrow$
          $LookAheadRoutine(\mu, down, LowerFix, UpperFix, SoftFix)$ ▷ Source of decrease is an unfixed college. Check if leads to a leximin increase;
  $Unfixed \leftarrow \{j | j \notin UpperFix$ or $(j, j') \notin SoftFix$ for some $j' > j\}$;

Note that the upper limit of a college c_j is fixed permanently when:

- Lowest ranked student matched to c_{j-1} causes a decrease in the leximin value on being matched c_j OR

- $up = j - 1$ and c_{j-1} causes a leximin value decrease.

This includes the case when $|\mu(c_{up})| = 1$. The lower limit of a college c_j is fixed when:

1. c_{j+1}'s upper limit is fixed,
2. $j = up$ and the valuation of c_j becomes less than that of the any unfixed college , OR
3. Giving out any more students would cause a leximin decrease, this includes the case when $|\mu(c_j)| = 1$.

We now prove the correctness of the algorithm.

Theorem 5.2 *FaSt-Gen (Algorithm 5.3) finds a leximin optimal stable matching given an instance of general ranked valuations in time* $O(m^2n^2)$.

Proof: We induct on $t \in [m]$, the rank of the colleges . For $j \in [m]$, let l_j and h_j indicate the ranks of the lowest and highest ranked students matched to c_j.

Base Case: t=1 When $t = 1$, $h_1 = 1$, that is, s_1 is the highest ranked student matched to c_1. This follows from Observation 5.2. Clearly, $s_{(l_1)+1} = s_{h_2}$. Recall that s_{h_2} was not included in $\mu(c_1)$ as it would be leximin inferior to giving it out. For any more students to be included in $\mu(c_1)$ would thus be leximin inferior, thus no leximin optimal matching would match c_1 to more students.

Now, c_1's lower limit was fixed because either: i) s_{l_1}'s valuation for c_2 would be too low, OR ii) c_1 could no longer give out anymore students without causing a leximin decrease. Consequently, any matching where s_{l_1} is not matched to c_1 will be leximin inferior to the current, restricted to the values of the first two colleges and $\{s_1, \cdots, s_{l_2}\}$, as in all of those matchings, the valuations of s_{l_1} and c_1 will always be strictly lower. Thus, c_1's lower limit is fixed correctly.

Induction Hypothesis: Let the matching of $c_1, \cdots, c_{t-1}$ be fixed as in an optimal matching. As a result, any matching which does not match $c_1, \cdots, c_{t-1}$ would be leximin inferior to those that do with respect to the matchings of the first t colleges and $\{s_1, \cdots, s_{l_t}\}$.

Now consider c_t. From Induction Hypothesis, c_t's upper limit is fixed correctly. There are now three possible cases:

Case 1: $t = m$. In this case, $l_t = l_m = n$ which is clearly correct.

Case 2: $t < m$ and at no point in the execution of the algorithm does $up = t$. In this case, c_t's lower limit has been fixed in Step 18 of FaSt-Gen. That is, s_{l_t}'s valuation for c_{t+1} was too low. Thus, any matching which matches s_{l_t} to a lower ranked college will be leximin inferior to the current one with respect to the matchings of the first $t+1$ colleges and $\{s_1, \cdots, s_{l_{t+1}}\}$.

Case 3: At some point in the execution, $up = t$. Analogous to the base case, changing the matching of c_t decreases the leximin value.

Hence, the lower limit of c_t is as in an optimal matching. □

Algorithm 5.4: LookAhead Routine

Input: $I, \mu, down, LowerFix, UpperFix, SoftFix$
Output: $\mu, LowerFix, UpperFix, SoftFix$

$\langle \mu', LF, UF \rangle \leftarrow \langle \mu,\ LowerFix,\ UpperFix \rangle$;
while $|LF| + |UF \backslash LF| < m$ **do**
  $up \leftarrow \min_{j \notin LowerFix} j$;
  if $|\mu(c_{up})| = 1$ *OR* $v_{up}(\mu) \leq v_{down}(\mu)$ **then**
    $LF \leftarrow LF \cup \{c_{up}\}$;
  else
    $\mu' \leftarrow Demote(\mu', up, down)$;
    if $\mathcal{L}(\mu') \geq \mathcal{L}(\mu)$ **then**
      $\mu \leftarrow \mu'$, $LoweFix \leftarrow LF$, $UpperFix \leftarrow UF$;
      break;
    else
      ▷ Decrease in leximin value, need to check the source;
      if $sourceDec(\mu', \mu) = c_{up}$ **then**
        $LF \leftarrow LF \cup \{c_{up}\}$, $UF \leftarrow UF \cup \{c_{up+1}\}$;
      else
        if $sourceDec(\mu', \mu) \in \mathcal{S}$ **then**
          $c_t \leftarrow \mu'(sourceDec(\mu', \mu))$;
          if $t = down$ **then**
            $UpperFix \leftarrow UpperFix \cup c_{down}$;
          else
            $SoftFix \leftarrow SoftFix \cup (down, t)$;
          break;

Look Ahead Routine: As with isometric valuations, it may be possible for a decrease in the leximin value to be made up for. We may encounter a setting where the valuation of a

demoted student decreases to the former valuation of c_{down}. The valuation of c_{down}, while it does increase, it is less than the student 's former valuation, causing a leximin decrease. We must decide if it is possible for c_{down} to make up for this decrease in valuation. Hence, we must look ahead till one of the following occurs: i) There is a leximin increase, ii) There is a leximin decrease due to a student OR iii) No more students can possibly be added to c_{down}'s matching.

In the first case, we must proceed with the new matching, and in the last we must revert to the old one, and fix c_{down}'s upper limit. In the case that there is a leximin decrease due to a student being matched to c_{down}, then clearly, we must revert to the old matching and fix c_{down}'s upper limit. The last possibility is that there is a leximin decrease due to a student matched to a higher ranked college . In this case, we have that so far our current matching continues to be leximin inferior to the old one, and for now, we cannot add any more students to c_{down}. Thus, similar to the case when this happens outside of a look ahead routine, we temporarily fix c_{down} till the offending college is open for giving out students.

Time Complexity: Note that without Observations 5.4 and 5.5, each leximin comparison takes $O(m+n) = O(n)$ time. Updating a matching takes $O(m) = O(n)$ time. Whenever a particular student - college is considered, a leximin comparison of the updated matching must be done. Pair (s_i, c_j) may be considered at most $j - 1 \leq m - 1$ times. Thus the algorithm considers mn pairs at most m times, each taking $O(n)$ time. Consequently, the time complexity is $O(m^2n^2)$. Even in the presence of capacity constraints, it is possible to find the leximin optimal stable matching in time $O(m^2n^2)$. The details of the algorithm are given in Appendix B.3

5.3.3 Strict Preferences and Constant Number of Colleges

We now show that when all agents' preferences are strict (but without the assumption of rankings) and there are exactly two colleges, even with an unrestricted number of students, we have a polynomial time algorithm to find the leximin optimal stable matching. Observe that, in this setting, when $m = 2$, any student s_i only forms a blocking pair with her most preferred college. As a result, for a matching μ to be stable, we need that for each student s_i, either s_i is matched to their favourite college (say c_j) or c_j prefers all the students matched to it under μ over s_i. A simple corollary of this is that the matching where all students are matched to their preferred college is stable. This is the student optimal stable matching.

Additional Notation

To this end, let us set up some notation for this specific case of $m = 2$. we use c_{-j} to denote the college other than c_j. As $m = 2$, we need to only know a student's most preferred college to determine their preference relation. To this end, We shall use the function α to capture the students' preference relations. We use $\alpha(s_i)$ to denote s_i's (most) preferred college and $-\alpha(s_i)$ to denote the college other than $\alpha(s_i)$. For $j = 1, 2$, we shall define the sets $A_j = \{s_i | \alpha(s_i) = c_j\}$.

Observe that A_1 and A_2 partition the set of students, and hence the complement of A_1, i.e. $\overline{A}_1 = A_2$ and similarly, $\overline{A}_2 = A_1$. For a matching μ, let $k_j(\mu)$ denote the number of students from A_j matched to c_j under μ, i.e., $k_j(\mu) = \mu(c_j) \cap A_j$. Now let $l_j(\mu)$ denote the number of the rest, i.e., $l_j(\mu) = |\mu(c_j) \setminus A_j| = |\mu(c_j)| - k_j$. Whenever, the matching being referred to is clear, we shall drop the (μ).

We shall use the functions top_j and $bottom_j$ to take as input a set of students and a number k and return college c_j's most preferred and least preferred k students from that set, respectively. Whenever $k > 1$, we shall use it as $top_j(S, , k)$, and when $k = 1$, we shall simply write $top_j(S)$.

Now for a matching μ to stable, for every $s_i \in A_j \setminus \mu(c_j)$, that is every student who prefers c_j but is not matched to it under μ, no student that c_j prefers less than s_i should be matched to it under μ. This means that, the students from $A_j \cup \mu(c_j) = top_j(A_j, k_j(\mu))$. Similarly, those not in A_j must be the l_j least preferred students of c_{-j}, that is, $\mu(c_j) \setminus A_j = bottom_{-j}(\overline{A}_j, l_j(\mu))$. Further, out of all these students from $\overline{A}_j$, c_j's least preferred one should still give c_j more value than the most preferred student from A_j matched to c_{-j}. otherwise, there will be a blocking pair. This is captured in the following observation.

Observation 5.6 *Given matchings instance $I = \langle \mathcal{S}, \mathcal{C}, U, V \rangle$ s.t. $m = 2$, a matching μ is stable if and only if the following holds. For each $j = 1, 2$, $\mu(c_j) = top_j(A_j, k_j) \cup bottom_{-j}(\overline{A}_j, l_j)$. Further, it also must hold that if for c_j, $l_j > 0$, then*

$$v_j(top_j(A_j \setminus \mu(c_j)))) < v_j(bottom_j(\mu(c_j) \setminus A_j)).$$

Iterating over stable matchings. The observation above is instrumental in enabling us to iteratively move over the space of stable matchings. We can start from any one stable matching and move to any other stable matching. We do this by toggling the matching of one student at a time, ensuring that all intermediate matchings are stable. This can be achieved as follows: Let the two matchings be μ_1 and μ_2. For each $j \in \{1, 2\}$ s.t. $k_j(\mu_1) < k_j(\mu_2)$,

toggle the matching of the students matched to c_j under μ_2 but not μ_1 in decreasing order of c_j's preference. That is, we toggle the matchings of students in $(\mu_2(c_j) \setminus \mu_1(c_j)) \cap A_j$. This toggle will not violate stability. After the toggle, for each college c_j such that it is matched to more students from $\overline{A}_j$ under μ_2 than μ_1, that is, $l_j(\mu_1) < l_j(\mu_2)$, bring in the students in $(\mu_2(c_j) \setminus \mu_1(c_j)) \cap \overline{A}_j$ in increasing order of c_{-j}'s preference, the resultant matching is μ_2. Consequently, we can go from any one stable matching to another without violating stability.

FaSt-Const Overview. We shall make use of this structure to find the leximin optimal stable matching. Our algorithm will proceed as follows: we first match all students to their preferred college. We then transfer out students from the rightmost college on the leximin tuple to the leftmost till we encounter one of the three possible settings:

1. leximin decrease due to a student,
2. violation in stability,
3. needing to undo a previous transfer.

All three of these conditions will be captured by a set $\mathcal{F}$ of forbidden pairs, where any student college pair whose being matched would cause any one of these three conditions are stored. We assume a function that toggles the matching of a student named $Toggle$ and it takes as input the student and the matching. We shall use μ_P to store the matching with the best leximin value discovered so far. We can use this as an alternate to having a separate lookahead procedure, as there are exactly two colleges.

Algorithm 5.5: FaSt-Const

Input: Instance $\langle \mathcal{S}, \mathcal{C}, U, V \rangle$ with $m = 2$
Output: μ_P

$\mu(s_i) \leftarrow \alpha(s_i)$ for each $i \in [n]$ ▷ Initialize to a stable matching;
$\ell \leftarrow \text{argmin}_{j=1,2}\, v_j(\mu)$, $r \leftarrow -\ell$;
$\mathcal{F} \leftarrow \{(s_i, -\alpha(s_i)) | u_i(-\alpha(s_i)) < v_\ell(\mu)\}$ ▷ Prevents irreversible leximin decrease;
$\mu_P \leftarrow \mu$;
while $(bottom_r(\mu(r)), \ell) \notin \mathcal{F}$ **do**
  $i^* \leftarrow bottom_r(\mu(r))$, $j^* \leftarrow r$;
  $\mu \leftarrow Toggle(\mu, s_{i^*})$;
  $\mu_P \leftarrow \text{argmax}\{\mathcal{L}_\mu, \mathcal{L}_{\mu_P}\}$;
  $\ell \leftarrow \text{argmin}_{j=1,2}\, v_j(\mu)$, $r \leftarrow -\ell$;
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{(s_i, -\alpha(s_i)) | u_i(-\alpha(s_i)) < v_\ell(\mu)\}$ ▷ Prevents irreversible leximin decrease
    $\cup \{(s_i, c_{j^*}) | v_{j^*}(s_{i^*}) \leq v_{j^*}(s_{i^*})\}$ ▷ Prevents stability violation and undoing this iteration

Note that students are moved from the rightmost college to the leftmost. Doing the opposite will always cause a decrease in leximin value at any time. We now show that the student-college pairs that are forbidden are never matched in a leximin optimal stable matching.

Lemma 5.2 *Given a matchings instance $I = \langle \mathcal{S}, \mathcal{C}, U, V \rangle$ with $m = 2$, let $\mathcal{F}$ be as in the end of Algorithm 5.5 when run on I. A leximin optimal stable matching on I does not match any s_i, c_j, s.t. $(s_i, c_j) \in \mathcal{F}$.*

Proof: Let $(s_i, c_j) \in \mathcal{F}$ and let μ be the matching when (s_i, c_j) are *first* added to $\mathcal{F}$. Three possible reasons are:

(i) $u_i(-\alpha(s_i)) < \min_{j=1,2} v_j(\mu)$.

This is only possible (for the first time when (s_i, c_j) are added to $\mathcal{F}$) if $\mu(s_i) = \alpha(s_i)$ and $c_j = -\alpha(s_i)$. Now observe that matching s_i to $-\alpha(s_i)$, will lead to a decrease in the value of s_i. Recall that s_i is the left most of the three agents whose values are changing. Thus, this will cause a leximin decrease. As a result, any matching μ' where $u_i(-\alpha(s_i)) < \min_{j=1,2} v_j(\mu')$, the leximin value will decrease by toggling s_i's matching to $-\alpha(s_i)$. The only way to undo this decrease, is to ensure that s_i is matched to $\alpha(s_i)$.

(ii) s_i is being moved from $\alpha(s_i)$ to $-\alpha(s_i)$.

Here, $c_j = \alpha(s_i)$, and s_i is being moved to $-\alpha(s_i)$. This implies that any other stable matching where s_i is matched to $\alpha(s_i)$ has either been explored or involves moving students from $-\alpha(s_i)$ (which is the leftmost college in the leximin tuple) to $\alpha(s_i)$. As a result, none of these matchings will be leximin superior to μ_P which stores the matching with the best leximin value out of the matchings explored. Consequently, moving s_i back to $\alpha(s_i)$, even later in the execution of the algorithm would either cause a stability violation, or undo multiple steps in the execution of the algorithm, none of which will lead to a higher leximin value than μ_P.

(iii) Another student $s_{i'}$ is being moved from $\alpha(s_{i'})$ to $-\alpha(s_{i'})$ and $v_{\alpha(s_{i'})}(s_i) < v_{\alpha(s_{i'})}(s_{i'})$.

Here, s_i is a student who while not being matched to c_j at the time of first forbidding (s_i, c_j), gives c_j less value than $s_{i'}$. $s_{i'}$ is the student being moved from c_j to c_{-j}. Now any stable matching where s_i is matched to c_j must also match $s_{i'}$ to c_j.v Thus, from case (ii), any such matching will not have leximin value greater than μ_p.

As the leximin optimal stable matching will have a leximin value at least as much as μ_P, the matching does not match any forbidden pairs from $\mathcal{F}$. □

Now, it suffices to show that for each student, if they were matched differently from what the algorithm returns, the matching would either be unstable or leximin inferior.

Theorem 5.3 *FaSt-Const (Algorithm 5.5) finds a leximin optimal matching given an instance with strict preferences and* $m = 2$ *in time* $O(n^2)$.

Proof: Let $I = \langle \mathcal{S}, \mathcal{C}, U, V \rangle$ with $m = 2$. Fix an arbitrary student s_i, for some $i \in [n]$ and let μ^* be the matching returned by FaSt-Const (Algorithm 5.5) and let $\mathcal{F}$ be as at the end of the execution of FaSt-Const on I. We consider three possible cases for the matching of s_i and show that in each matching s_i differently would have led to either an unstable matching or one that is leximin inferior.

Case 1: $\mu^*(s_i) = c_j$ and $(s_i, c_{-j}) \in \mathcal{F}$

From Lemma 5.2, we have that any matching that matches s_i to c_{-j} is either unstable or leximin inferior to μ^*.

Case 2: $\mu^*(s_i) = \alpha(s_i)$ and $\{(s_i, \alpha(s_i)), (s_i, -\alpha(s_i))\} \cup \mathcal{F} = \emptyset$.

This implies that s_i was never considered for $-\alpha(s_i)$ during the entire execution of the algorithm. This is possible in one of two ways:

1. $\alpha(s_i)$ is the leftmost college in the leximin tuple of μ^*. Hence, any stable matching μ where s_i is not matched to would either match a forbidden pair or give $\alpha(s_i)$ even lower value than in μ^*, making μ leximin inferior to μ^* OR

2. $\alpha(s_i)$ is the rightmost college in the leximin tuple of μ^*. Let $s_{i'} = Bottom(\mu^*(\alpha(s_i), \alpha(s_i))$. Thus, we have that $(s_{i'}, -\alpha(s_i)) \in \mathcal{F}$. Now any stable matching μ which matches s_i to $-\alpha(s_i)$ also matches $s_{i'}$ to $-\alpha(s_i)$, which are forbidden, hence μ will be leximin inferior to μ^*.

Case 3: $\mu^*(s_i) = \alpha(s_i)$, $(s_i, -\alpha(s_i)) \notin \mathcal{F}$ but $(s_i, \alpha(s_i)) \in \mathcal{F}$.

This implies that s_i was considered for $-\alpha(s_i)$ during the execution of FaSt-Const but the matching was lower in leximin value to μ^*. Thus, till the end of the execution of FaSt-Const, the leximin value of the matchings considered were not higher than μ^*, and the execution stopped due to some other forbidden pair. This implies that any stable matching where s_i

is matched to $-\alpha(s_i)$ which has not been explored by during the execution of the algorithm requires matching a forbidden pair or sending students from the leftmost college to the rightmost. In either case the matching will be lower in leximin value to μ^*. This completes the proof.

□

Time Complexity: Observe that each iteration of the while loop toggles the matching of one student at a time and no student is ever moved twice. Within each step takes at most time $O(n)$, as a result, FaSt-Const takes time at most $O(n^2)$.

We now show that in more general settings, especially in the absence of strict rankings, the problem of finding the leximin optimal stable matching becomes intractable

5.4 Intractability without Strict Rankings

We shall now establish the necessity of strict preferences and rankings by establishing hardness otherwise. We shall set $b_j = n - m + 1$ in these reductions. We first look at a strict preferences setting without rankings. We find a reduction from the Subset Sum problem (SSP) to this, making it NP-Hard. In SSP, given a set of k positive integers, $A = \{a_1, \cdots, a_k\}$, and a target integer value B, such that $\max_A a_i < B \leq \sum_A a_i$, we must decide whether there exists a subset $S \subseteq A$, such that $\sum_S a_i = B$. This is a well-known NP-Hard problem [70].

Theorem 5.4 *It is NP-Hard to find a leximin optimal stable matching under strict preferences.*

Proof: Given an instance of SSP with $A = \{a_1, \cdots, a_k\}$ and B we construct an instance of stable many-to-one matchings (SMO) as follows.

Setup. Set the number of colleges $m = k + 1$ and the number of students $n = 2k$. We shall construct one student s_i for each integer a_i. The most preferred college of s_i will be c_i. We shall define c_m to be the college whose matching (set of students) will correspond to the subset selected for SSP. The remaining students ensure that all the colleges c_j which are not matched to their corresponding s_j are not unmatched. These students will be $s_{k+1}, \cdots, s_{2k}$. For each $j \in [k]$, the most preferred college of s_{j+k} will be c_j. For stability, the most preferred student of c_j, for $j \in [k]$ is s_{j+k}, the second most preferred is s_j. Consequently, with $\epsilon = \frac{1}{3k^2} < \frac{1}{n}$, the valuations are defined as follows:

For student s_i, $i \in [k]$ if $i = j$, set $u_i(c_j) = B$. If $j = m$, set $u_i(c_j) = B - a_i + \epsilon$, else set $u_i(c_j) = j\epsilon$. For student s_i, $i \in [2k] \setminus [k]$, if $i = j + k$, set $u_i(c_j) = B$, else, $u_i(c_j) = j\epsilon$. For college c_j, with $j \in [k]$, if $i = j + k$, set $v_j(s_i) = 2B$, if $i = j$, define $v_j(s_i) = B$ else, $v_j(s_i) = i\epsilon$.

Finally, for the college corresponding to the subset selected, if $i \in [k]$, define $v_m(s_i) = a_i$, else, $v_m(s_i) = i\epsilon$.

Properties of the instance constructed. Observe that $j\epsilon < 1 < B - a_i + \epsilon < B$ for all $i \in [k], j \in [n]$ and hence, all preferences are strict. As a result, for each s_i for $i \in [k]$, the most preferred college is c_i and the second most preferred college is c_m. Further, in any complete *stable* matching μ, we have that for $i \in [2k] \setminus [k]$, $\mu(s_i) = c_{i-k}$. If this doesn't hold for some $i \in [2k] \setminus [k]$, then $s_i - c_{i-k}$ is a blocking pair.

For $i \in [k]$, for any complete stable matching μ, if $\mu(s_i) \notin \{c_i, c_m\}$, μ is leximin dominated by the matchings which are identical to μ except s_i is matched to one of c_i or c_m. Thus, we shall henceforth only consider stable matchings μ where for $i \in [k]$, $\mu(s_i) \in \{c_i, c_m\}$ and for $i \in [2k] \setminus [k]$, $\mu(s_i) = c_{i-k}$. Let μ^{i+} denote the matching that is identical to μ except that s_i is matched to c_i, i.e. if $i' \neq i$, $\mu^{i+}(s_{i'}) = \mu(s_{i'})$, and $\mu^{i+}(s_i) = c_i$. Similarly define μ^{i-} s.t. if $i' \neq i$, $\mu^{i-}(s_{i'}) = \mu(s_{i'})$, and $\mu^{i-}(s_i) = c_m$. Observe that if μ is stable, so are μ^{i+} and μ^{i-}.

Leximin domination among stable matchings. We shall now show that the leximin optimal stable matching will try to ensure that c_m gets value B. To this end, we consider arbitrary, complete stable matchings and show when would moving one student , say s_i either to c_m or c_i improve the leximin value. To show that one matching leximin dominates another, here, we need only consider the agents whose values change: s_i, c_i and c_m, and show that the minimum of these agents' values is more in one matching than the minimum of these agents' values in the other.

Now fix a complete stable matching μ. If c_m's value for μ is greater than B, that is $v_m(\mu) > B$, then for any student $s_i \in \mu(c_m)$, they lie to the left of c_m in the leximin tuple with value $u_i(\mu) = B - a_i + \epsilon < B < v_m(\mu)$. Now, it is easy to see that $v_m(\mu^{i+}) = v_m(\mu) - a_i \geq B + 1 - a_i > u_i(\mu)$. The valuations of all other agents remain unchanged or increase. As a result, if $v_m(\mu) > B$, then for any $s_i \in \mu(c_m)$, $\mathcal{L}_{\mu^{i+}} > \mathcal{L}_\mu$.

On the other hand if $v_m(\mu) \leq B$, then for any $s_i \in \mu(c_m)$, $u_i(\mu) = B - a_i + \epsilon > B - a_i \geq v_m(\mu) - a_i = v_m(\mu^{i+})$. Thus, μ is leximin superior to μ^{i+} for any $s_i \in \mu(B_m)$. For any $s_i \notin \mu(c_m)$ such that $a_i \leq B - v_m(\mu)$, we find that $u_i(\mu^{i-}) = B - a_i + \epsilon > B - a_i \geq v_m(\mu)$. As a result, $\mathcal{L}_{\mu^{i-}} > \mathcal{L}_\mu$ for any $s_i \notin \mu(c_m)$ such that $a_i \leq B - v_m(\mu)$. Now for s_i such that $a_i > B - v_m(\mu)$, $u_i(\mu^{i-}) = B - a_i + \epsilon \leq B - (B - v_m(\mu) + 1) + \epsilon < v_m(\mu)$. Thus, $\mathcal{L}_{\mu^{i-}} < \mathcal{L}_\mu$.

Consequently, the leximin optimal stable matching μ^* will always maximize $v_m(\mu)$ with the constraint that $v_m(\mu^*) \leq B$ and will match c_m to the set of students corresponding to the subset $S^* = \text{argmax}_{S \subseteq A, \sum_S a_i \leq B} \sum_S a_i$. Thus, the required subset exists if and only if $v_m(\mu^*) = B$. □

As a result, for unrestricted m, with strict preferences, without rankings the problem of finding a leximin optimal remains intractable. In fact, the preferences of the agents in the instances created in the above proof are actually very similar with a few changes only. We now find that under isometric valuations and weak rankings (strict preferences need not be satisfied) , finding the leximin optimal stable matching is NP-Hard, even with a constant number of colleges.

Theorem 5.5 *It is NP-Hard to find the leximin optimal stable matching under isometric valuations with weak rankings with $m = 2$ and strongly NP-Hard with $n = 3m$.*

Proof: For the $m = 2$ setting, we give a reduction from the balanced partition problem, which is known to be NP-Complete [70].

Balanced Partition Problem: Given a set of integers $P = \{p_1, \cdots, p_k\}$, such that $\sum_{i=1}^{k} p_i = 2B$, find a 2-partition $\{A_1, A_2\}$ of P which satisfies $\sum_{p_i \in A_j} p_i = B$ for all $j \in \{1, 2\}$.

Given P, we shall construct an instance of isometric valuations I with $m = 2$, such that P admits a balanced partition if and only if the leximin optimal stable matching of the instance I allocates valuation of B to both c_1 and c_2. Set $n = k$, $V_{ij} = p_i$ for all $i \in [n], j \in \{1, 2\}$. In such a setting it is easy to see that all complete matchings are stable as no student has any incentive to deviate. In any matching, students will always get the same value, that is student s_i always gets value p_i. Thus for a leximin optimal matching, it suffices to check the values that the colleges attain.

Let P admit a balanced partition $\{A_1, A_2\}$. Let μ be the matching which matches c_j to all the students whose values are in A_j for all $j = 1, 2$. Here, clearly, all colleges get value B. The leximin tuple of μ will list the p_i values first, in non-decreasing order and the last two entries will all be B. Any matching that gives any one of the colleges, say c_1, higher valuation, will naturally decrease the valuation of the another college. Hence c_2's value will either be lower in the same position in the leximin tuple, or be to the left, resulting in a lower leximin value in both cases. Hence, μ is a leximin optimal matching.

Conversely, if the leximin optimal matching gives value B to all colleges then the partition created by the matching is clearly the required balanced partition.

For the $n = 3m$ case we give a reduction from the 3-Partition problem which is known to be strongly NP-Hard[70].

3-Partition Problem: Given a set of integers $P = \{p_1, \ldots, p_k\}$, such that $\sum_{i=1}^{k} p_i = {}^{3t}/_{k}$, find a ${}^{k}/_{3}$-partition $\{A_1, \cdots, A_{k/3}\}$ of P which satisfies $\sum_{p_i \in A_j} p_i = t$ for all $i \in [{}^{k}/_{3}]$.

Given an instance of the 3-partition problem P, we create a matchings instance as follows. Set $m = {}^{k}/_{3}$, $n = k$ and set $V_{ij} = p_i$ for all $j \in [m]$. Observe that this is identical to the previous construction, with the exception that there are now more colleges. Thus, from the same reasoning, for a leximin optimal matching, it suffices to check the values that the colleges attain.

Let P admit a 3-partition $\{A_1, A_2, \cdots, A_{n/3}\}$. Let μ be the matching which matches c_j to all the students whose values are in A_j for all $j \in [m]$. Here, clearly, all colleges get value t. The leximin tuple of μ will list the p_i values first, in non-decreasing order and the last m entries will all be t. Any matching that gives any one of the colleges, say c_j, higher valuation, will naturally decrease the valuation of the another college, say $c_{j'}$. Hence $c_{j'}$'s value will either be lower in the same position in the leximin tuple, or be to the left, resulting in a lower leximin value in both cases. Hence, μ is a leximin optimal matching.

Conversely, if the leximin optimal matching gives value t to all colleges then the partition created by the matching is clearly the required 3-partition. □

5.4.1 Hardness of Approximation

We now discuss the possibility of finding some approximation to the leximin optimal stable matching. We shall show that under weak rankings, unless P=NP, no polynomial factor approximation is possible. We first define what an approximation algorithm for finding a leximin optimal (over any space) must guarantee.

Definition 5.5 (α-approximation of leximin) *An algorithm is to give an α-approximation $\alpha \in (0, 1)$ to the leximin optimal, if given instance I with μ^* as the leximin optimal solution, it outputs μ s.t. for each index t of the leximin tuple, $\alpha\mathcal{L}_{\mu^*}[t] \leq \mathcal{L}_{\mu}[t] \leq \frac{1}{\alpha}\mathcal{L}_{\mu^*}[t]$*

This is a generalization of the definition of approximation algorithms which optimize for a single value. Note that it important that the approximation factor holds for every index. We would not like a setting where the first index of the leximin tuple satisfies the condition but the subsequent indices do not. This is because a good approximation to the first index may be satisfied by all stable matchings and would not guarantee any fairness to any of the other agents. We now give a reduction from the Bin Packing problem to establish the hardness of approximation for leximin optimality. We look at the decision version of the problem where given a set of items G, each with weight w_i and k bins of capacity 1, we must decide if there

is a partition of the items into k bins such that the sum of the weights of the items in each bin is at most 1. We assume, without loss of generality, that $w_i \leq 1$ and $|G| \geq k > 1$.

Before proving Theorem 5.6, we will develop some of the necessary machinery for its proof in the following lemma.

Lemma 5.3 *Unless P=NP, no* $1/2 + \epsilon$ *approximation algorithm exists for finding the leximin optimal stable matching*

Proof: Given a bin packing instance $\langle G, k, \{w_i\}_{g_i \in G} \rangle$, where $0 \leq w_i \leq 1$ for all items $g_i \in G$, $\sum_{g_i \in G} w_i \leq k$, and $|G| > k$, we construct an instance of stable many-to-one matching (SMO) as follows.

Setup. Set $n = |G| + 1$, we create a student for each item and an additional dummy student. Further, set $m = k + 1$, we create a college for each bin and an additional dummy college which all the students will prefer to the bins. We shall interchangeably refer to these non-dummy students and colleges as items and bins. The valuations are then set so that the following properties hold:

P1. All the students $s_1, \cdots, s_{n-1}$ prefer the dummy college c_n over other colleges,

P2. All the non-dummy colleges (bins) $c_1, \cdots, c_{m-1}$ have no value for the dummy student s_n,

P3. The dummy student s_n and dummy college c_m are always matched in a leximin optimal stable matching, and

P4. At least one of the non-dummy students (items) $s_1, \cdots, s_{n-1}$ will be matched to the dummy college c_m if and only if there is no bin packing.

We set the valuations as follows: Each non-dummy student or item's valuation towards the non-dummy colleges or bins are equal and given by $u_i(c_j) = 1 - w_i + \epsilon$, for each $i \in [n-1]$ and each $j \in [m-1]$, where, $\epsilon > 0$ is an infinitesimally small value. Moreover, to ensure that the students prefer the dummy college c_m over the bins, we set $u_i(c_m) = 2$ for each $i \in [n-1]$. For the dummy student s_n, we set $u_n(c_j) = 0$ for all $j \in [m-1]$ and $u_n(c_m) = 1$.

The bins' valuations for each item i is the weight w_i, i.e, $v_j(s_i) = w_i$ for each $j \in [m-1]$ and each $i \in [n-1]$. Also, for any bin, the value for the dummy student is $v_j(s_n) = 0$. For the dummy college c_m, we set $v_m(s_i) = n$ for all $i \in [n]$. If s_n is not assigned to c_n, the valuation

obtained by s_n would be 0. Hence, the lexicographically ordered valuations obtained by assigning s_n to c_m would always dominate all such matchings where all other agents are the same but s_n is assigned to a bin. Hence, a leximin optimal stable matching will always match s_n to c_m.

We now show that, in the leximin optimal stable matching, each item is assigned to some bin if and only if a bin packing exists.

Finding the leximin optimal stable matching from a bin packing. Suppose the bin packing instance $\langle G, k, \{w_i\}_{g_i \in G} \rangle$ admits a bin packing. We claim that the leximin optimal stable matching, matches all the items to bins. Let μ be the leximin optimal over the space of all stable matchings where the items are matched to bins only. Note that μ is also the solution obtained by taking leximin optimal of bin valuations over all bin packing solutions. Thus, for each bin c_j, we have $v_j(\mu) \leq 1$.

Now, fix an item s_i, $i \in [n-1]$. As s_i is matched to a bin, say c_j, under μ, $u_i(\mu) = 1 - w_i + \epsilon$. Now let μ' be identical to μ, with the exception that s_i is matched to c_m. As a result, the value of c_j for μ' is $v_j(\mu') = v_j(\mu) - w_i \leq 1 - w_i < u_i(\mu)$ and valuation of s_i would increase to 2, and $v_m(\mu') = 2n$. The values of all other agents are identical in μ and μ'. Thus μ' is leximin inferior to μ. To overcome this, we must take an item $s_{i'}$ from another bin, which decreases its valuation, in turn to less than $1 - w_{i'} + \epsilon$, thus this matching is also leximin inferior to μ. Hence, every time we attempt to cover for the decrease in the leximin value, will create a new bin whose valuation is less than the valuation of the item removed from it. This will always be leximin inferior to μ. Consequently, if a bin packing exists, the leximin optimal matching must match the items to the bins only, and hence, c_m's valuation will be n.

Constructing a bin packing from leximin optimal stable matching. Now suppose that no bin packing solution exists, any matching that matches all the items to bins only, will give at least one bin valuation strictly greater than 1. Consider any item matched to this bin. Matching it to c_m will reduce the bin's valuation but it will still remain greater than $1 - w_i + \epsilon$, s_i's current valuation (ϵ is infinitesimally small). Along with this, the valuation of s_i and c_m will both increase and the remaining agents' will not be affected. As a result, whenever a bin packing does not exist, at least one item will be matched to c_m, and thus c_m's valuation will be at least $2n$. This shows that the valuations ensure that properties P1-P4 are true.

Suppose, for some $\alpha > 1/2$, an α-approximation algorithm, say ALG, exists for finding the leximin optimal stable matching. Now, let μ' be the matching that ALG outputs on the matching instance I corresponding to the bin packing problem and let μ be the leximin optimal stable matching. Recall that by construction of I, c_m's valuation always appears

highest (last) in the leximin tuple. Therefore, whenever a bin packing exists $v_m(\mu) = n$ and thus, $\alpha n < v_m(\mu') < \frac{n}{\alpha}$. Since $\alpha > 1/2$, $\frac{n}{2} < v_m(\mu') < 2n$. Moreover, if a bin packing does not exist, $v_m(\mu) \geq 2n$, and thus, $v_m(\mu') > n$. Further, by construction, $v_m(\cdot)$ can only have values which are integral multiples of n, thus $v_m(\mu') \geq 2n$ whenever a bin packing does not exist.

Hence, unless P=NP, no $\frac{1}{2} + \epsilon$-approximation algorithm exists, for any $\epsilon > 0$. □

This technique can be extended using a more intricate reduction to establish the following theorem.

Theorem 5.6 *Unless P=NP, for any $\delta > 0$, $c \in \mathbb{Z}^+$ there is no $1/cn^\delta$-approximation algorithm to find a leximin optimal stable matching under unconstrained additive valuations.*

Proof: This proof builds upon the proof of Lemma 5.3. In order to get a better bound for the hardness of approximation, we must increase the gap in c_m's valuations while ensuring that c_m's valuation always appears at the end of any complete[1] stable matching. We do this by replicating the bin packing instance.

Thus, for any $\delta > 0$, $c \in \mathbb{Z}^+$, we can set $t = \lceil cn^\delta \rceil$. Now, given a bin packing instance $\langle G, k, \{w_i | g_i \in G\}\rangle$, we create an SMO instance as follows. Let $|G| = l$.

Setup. Set $n = tl + 1$ and $m = tk + 1$ and hence, replicate the bin packing instance t times. That is, for each item we create t copies of it and now there are tk bins. We shall keep these copies from interacting with each other, that is, items from one copy will only be matched to bins from the same copy. The valuations will continue to satisfy the properties P1-P4 listed in the proof of the previous lemma. The valuation functions are accordingly defined as follows:

For the p^{th} copy of bin j, $c_{(p-1)k+j}$, it has value for items from the same or any earlier copy, hence we set $v_{(p-1)k+j}(s_{(p'-1)l+i}) = w_i$ for all $j \in [k]$, $p, p' \in [t]$, $i \in [l]$ s.t. $p \geq p'$ and $v_{(p-1)k+j}(s_{(p'-1)l+i}) = 0$ for all $j \in [k]$, $p, p' \in [t]$, $i \in [l]$ s.t. $p < p'$. None of the bins have value for the dummy student s_n, so we set $v_j(s_n) = 0$ for all $j \in [m-1]$. For the dummy college c_m, all students give equal value and we set $v_m(s_i) = (t+1)n$ for all $i \in [n]$.

For the p^{th} copy of item i, $s_{(p-1)l_i}$, it has no value for bins from later copies and equal value for all bins from the same or earlier copies. To this end, we set $u_{(p-1)l+i}(c_{(p'-1)k+j}) = 1 - w_i + \epsilon$ for all $i \in [l], p, p' \in [t], j \in [k]$ s.t. $p \geq p'$ and $u_{(p-1)l+i}(c_{(p'-1)k+j}) = 0$ for all $i \in [l], p, p' \in [t], j \in [k]$ s.t. $p < p'$. All items have the highest value for the dummy college c_m, so we set $u_i(c_m) = t + 1$ for all $i \in [n-1]$. The dummy student s_n has value only for the dummy college c_m, consequently, we define $u_n(c_j) = 0$ for all $j \in [m-1]$ and $u_n(c_m) = 1$. These are summarised as follows.

[1]Recall that we say a matching is complete if each agent's matching is non-empty

$$v_{(p-1)k+j}(s_{(p'-1)l+i}) = w_i \qquad \text{for all } j \in [k],\ p,\ p' \in [t],\ i \in [l] \text{ s.t. } p \geq p' \tag{5.1}$$

$$v_{(p-1)k+j}(s_{(p'-1)l+i}) = 0 \qquad \text{for all } j \in [k],\ p, p' \in [t],\ i \in [l] \text{ s.t. } p < p' \tag{5.2}$$

$$v_j(s_n) = 0 \qquad \text{for all } j \in [m-1]$$

$$v_m(s_i) = (t+1)n \qquad \text{for all } i \in [n]$$

$$u_{(p-1)l+i}(c_{(p'-1)k+j}) = 1 - w_i + \epsilon \qquad \text{for all } i \in [l], p, p' \in [t], j \in [k] \text{ s.t. } p \geq p' \tag{5.3}$$

$$u_{(p-1)l+i}(c_{(p'-1)k+j}) = 0 \qquad \text{for all } i \in [l], p, p' \in [t], j \in [k] \text{ s.t. } p < p' \tag{5.4}$$

$$u_i(c_m) = t+1 \qquad \text{for all } i \in [n-1]$$

$$u_n(c_j) = 0 \qquad \text{for all } j \in [m-1]$$

$$u_n(c_m) = 1$$

The construction is similar to the one in the proof of Lemma 5.3. Here, we make t copies of each item and bin so that we get t copies of the bin packing instance in the constructed instance. Further, for each $p \in [t]$, $s_{(p-1)l+1}, \cdots, s_{(p-1)l+l}$ represent $g_1, \cdots, g_n$ respectively. Also, $c_{(p-1)k+1}, \cdots, c_{(p-1)k+k}$ represent the k bins of capacity 1. For each $p \in [t]$, we consider $s_{(p-1)l+1}, \cdots, s_{(p-1)l+l}$ and $c_{(p-1)k+1}, \cdots, c_{(p-1)k+k}$ to represent the p^{th} copy.

Properties of the instance constructed. The valuations are defined such that all agents in a copy p have value 0 for all agents in copy p' for all $p' > p$ (from Equations 5.2 and 5.4). Consequently, no leximin optimal matching will match a student in copy p to a college in copy p' when $p' \neq p$, as this would mean this student gets value 0. Further, from Equation 5.3, for all $p \in [t]$, each student in copy p, is indifferent between all the colleges in copies $1, \cdots, p$. However, the colleges in copies $1, \cdots, p-1$ have valuation 0 for students in copy p (from Equation 5.2), ensuring that all items are matched to bins in their own copy or c_m.

Thus, from an analogous argument to that in the proof of the previous lemma, if a bin packing exists, the items are matched to bins (of the same copy) in the leximin optimal stable matching. Thus, it gives c_m is matched only to s_n and thus has a valuation of exactly $(t+1)n$. If a bin packing doesn't exist, at least one item from each copy is matched to c_m giving it a valuation of at least $(t+1)^2 n$.

We shall now show that a $1/cn^\delta$-approximation algorithm with $\delta > 0$ and $c \in \mathbb{Z}^+$ to find a leximin optimal stable matching with weak rankings will match c_m to at least one item if and only if a bin packing doesn't exist. Recall that we set $t = \lceil cn^\delta \rceil$. Now let a $1/cn^\delta$-approximation algorithm exist, call it ALG. Further, let μ^* denote a leximin optimal stable matching of the instance constructed. Then clearly $u_i(\mu^*) > 0$ and $v_j(\mu^*) > 0$ for all $i \in [n]$

and $j \in [m]$.

Further note that by construction, c_m's valuation will always appear in the last index of the leximin tuple of any matching where c_m is matched with at least one student . As $1/cn^\delta > 0$, $v_m(\mu_{ALG}) \geq (t+1)n$ and it will be the last index in the leximin tuple of μ_{ALG}. Similarly for the dummy student s_n, $u_n(\mu_{ALG})$¿0. Consequently, $\mu_{ALG}(s_n) = c_m$. Further, there do not exist $p, p' \in [t]$, $i \in [l]$ and $j \in [k]$ s.t. $p \neq p'$ and $\mu_{ALG}(s_{(p-1)l+i}) = c_{(p'-1)k+j}$.

Constructing a leximin optimal stable matching from a bin packing. Now let a bin packing exist. Hence, for all students, that is for all $i \in [n]$, $0 < u_i(\mu^*) \leq 1$ and for all bins $j \in [m-1]$ $0 < v_j(\mu^*) \leq 1$. Thus, for all $d \in [m+n-1]$, $\mathcal{L}_{\mu^*}[d] \leq 1$. Further, $v_m(\mu^*) = (t+1)n$.

Now as c_m's valuation will be the greatest in μ_{ALG}, for any student s_i, $i \in [n-1]$, $u_i(\mu_{ALG})$ must appear in the first $m+n-1$ entries of $\mathcal{L}_{\mu_{ALG}}$, say d_i. For $i \in [n-1]$, let d_i be index where s_i's value lies in the leximin tuple for μ_{ALG}. Thus, we have that $u_i(\mu_{ALG}) = \mathcal{L}_{\mu_{ALG}}[d_i] \leq cn^\delta \mathcal{L}_{\mu^*}[d_i] \leq cn^\delta < t+1$

Thus, whenever there is a bin packing, no s_i, for $i \in [n-1]$, is matched to c_m. Consequently, the dummy college gets value $v_m(\mu_{ALG}) = (t+1)n$.

Properties of leximin optimal stable matchings when no bin packing exists. If a bin packing does not exist, we have that c_m is matched to at least one item from each of the t copies under μ^* and $v_m(\mu^*) \geq t(t+1)n$. Further, as c_m is the last index of the leximin tuple in both μ^* and μ_{ALG}, we have that, $v_m(\mu_{ALG}) \geq (1/cn^\delta)t(t+1)n$. Now as $t = \lceil cn^\delta \rceil$, we have that

$$v_m(\mu_{ALG}) \geq \frac{t(t+1)n}{t} > (t+1)n.$$

Thus, a bin packing exists if and only if $v_m(\mu_{ALG}) = (t+1)n$. □

Note that the above reduction constructs an instance with weak rankings. Thus, even when the instances have weak rankings, the hardness of approximation remains. This clearly subsumes the case when there are no consistent rankings. This concludes all the results of this chapter.

5.5 Conclusion

This chapter initiated the study of leximin fairness and stability in many-to-one matchings. We showed that strict rankings are essential to find the leximin optimal over stable matchings in polynomial time when there is no restriction on the number of colleges and students. We showed that on relaxing either strict preferences or rankings, the problem becomes computationally intractable. With only strict preferences, when $m = 2$, we found an algorithm which finds the leximin optimal stable matching in time polynomial in n. One area of future

work would be to see if with strict preferences, an XP algorithm parameterized by the number of colleges is possible. Other fairness notions like Nash welfare and p-means can also be explored. A more comprehensive discussion is in Chapter 7.

Chapter 6

Stability and Incentive Compatibility in Fractional Matchings

Incentive compatibility is extremely desirable in most practical situations. This chapter studies the incentive compatibility of mechanisms finding stable fractional matchings. We exhibit matching instances for which no stable fractional matching mechanism is approximately incentive compatible. Then, we characterize the class of matching instances with unique stable fractional matchings. For a given matching instance to have a unique stable fractional matching, we establish that it is necessary and sufficient for it to satisfy the conditional mutual first preference (CMFP) property. To this end, we provide an algorithm that uses envy graphs finding a non-integral stable matching whenever the preferences are strict and the given instance is not in CMFP. For this class of CMFP matching instances, we prove that every mechanism that produces the unique stable fractional matching is incentive compatible.

6.1 Introduction

Even within two-sided matching literature, one-one matchings are well studied [52, 101, 114, 115, 119]. In these familiar settings, including the previous chapters, typically, the nodes on the two sides are wholly or "integrally" matched (these are called integral matchings). A fractional matching is a convex combination of integral matchings and thus generalizes integral matchings in a natural way. Fractional matchings have largely been studied in the literature only as a means to produce integral matchings. Even the majority of papers that explicitly study fractional matchings only study them to gain a deeper understanding of inte-

gral matchings, for example [103, 109, 114]. Only recently, Caragiannis et al. [36] presented an exclusive study on the space of stable fractional matchings. Incentive compatibility is another key requirement in the context of fractional matchings but has not been explored. This work reports the first investigation into the important topic of incentive compatibility of matching mechanisms to find stable fractional matchings. We focus on fractional matchings with cardinal values.

Fractional Matchings

There are many practical situations where fractional matchings are relevant. Consider for instance, labour markets. Here freelancing experts or professionals can spend different fractions of their time working for multiple organizations. Labour markets have been mentioned in Caragiannis et al. [36]. One can think of many other applications, for example, matching markets for cloud space. Here, it is not necessary for data to be stored entirely on one server. Another application is procurement markets for matching wholesale buyer with different wholesale sellers. Wholesale buyers have large demands and can procure different fractions of their required items from different sellers.

Exploring fractional matchings to investigate various properties studied in the context of integral matchings could raise interesting challenges. Relaxing the integrality constraint in matchings may make the problem of finding matchings that satisfy desirable properties harder. Consider, for instance, the problem of maximizing social welfare amongst all *stable* matchings. For integral matchings, this can be posed as a linear program. However, when we allow for matchings to be fractional, the problem becomes APX-Hard, as shown by Caragiannis et al. [2021]. Caragiannis et al. also show that by allowing the stable matchings to be fractional, we can make large gains in terms of social welfare. It is therefore of interest to study fractional matchings and devise efficient algorithms to find fractional matchings with desirable properties.

Incentive compatibility is clearly a desirable property to seek. For many real wold application, it is desirable for the matching mechanism that produces the matching to induce all participating agents to report their true preferences. A considerable amount of work has gone into studying the incentive compatibility properties of various algorithms, particularly of the Gale-Shapley algorithm [101, 110, 115, 119, 123, 126]. In this chapter, we investigate the important but unexplored problem of finding incentive compatible mechanisms to produce stable fractional matchings. In doing so, we characterize the space of matching instances with a unique stable fractional matchings. This is of independent interest in the context of structural properties of stable fractional matchings.

6.1.1 Technical Contributions of this Chapter

As already stated, our focus in this chapter, is on fractional matchings. These settings may have agents with weak preferences or strict preferences. While most of the results in this chapter hold for the case of weak preferences, one particular result (namely concerning Algorithm 2) holds only for strict preferences. We will be defining these terms more formally in Section 6.2.1. Section 6.2.3 provides relevant structural observations on the space of stable fractional matchings. Following are our contributions on the topic of incentive compatibility of matching mechanisms to find stable fractional matchings.

- First, in Section 6.3, we make a significant observation: that there are matching instances for which *no mechanism* that produces a stable fractional matching is incentive compatible. Further, for those instances where for any agent, the value from the mechanism is greater than half the value from misreporting, no mechanism can always produce a stable matching. This motivates us to seek restricted classes of matching instances where an incentive compatible mechanism exists, that produces a stable fractional matching.

- We characterize, in Section 6.3.2, restricted classes of matching instances that admit a *unique* stable fractional matching. Specifically, we show that a unique stable fractional matching will exist if and only if the given matching instance satisfies the conditional mutual first preference property (CMFP).

- For the above class of matching instances, we show that *every* mechanism that produces the unique stable fractional matching will be incentive compatible. Furthermore, the unique fractional matching will be resistant to *coalitional manipulations*.

- We next provide, in Section 6.4, an efficient algorithm (Algorithm 6.2) that constructs, under strict preferences, stable fractional matchings that are not integral. Our algorithm makes intelligent use of envy graphs, hitherto unused in the stable matchings literature.

The above algorithm may not work if the preferences are weak. All other results above hold even if the preferences are weak. While the focus of this chapter is on incentive compatible mechanisms, the characterization of the space of stable matching instances with a unique stable fractional matching is of independent interest. The proof of this characterization also leads to an interesting corollary about the number of stable fractional matchings possible under strict preferences. Before presenting our results, we first discuss the literature relevant to this chapter.

6.1.2 Prior Relevant Work

Fractional Matchings in Prior Work

Exploration of fractional matchings has been limited in the past. Most of the work in this space comes from prior work investigating the structure of the space of stable matchings. Work by Roth et al. [103] and Teo and Sethuraman [114] used linear programming formulations to capture and analyze the stable marriage problem. Both these works established the integrality of the stable matchings polytope. The fractional matchings studied in both the papers are said to be stable if the closest integral matching is stable. In general, in the majority of literature, a fractional matching has been considered as stable if the matchings in its support are stable. Most matching algorithms that find fractional matchings use them for rounding to an integral matching.

Stable Fractional Matchings

A key roadblock in the initial analysis of fractional matchings was that the preferences considered were ordinal and not cardinal. Consequently, there was no non-trivial way to define the stability of a fractional matching by itself[1]. Caragiannis et al. [36] overcame this roadblock by considering a stable matching setting with cardinal values. The focus of Caragiannis et al. [36] is to consider the social welfare of stable fractional matchings. In many instances, fractional matchings are able to achieve higher social welfare when compared to integral matchings. Consequently, the aim of Caragiannis et al. [36] was to find a stable fractional matching which maximizes social welfare. This problem was called SMC. The authors make a series of structural observations about the space of stable fractional matchings (such as non-convexity of this space [2]), provide an efficient approximation algorithm and two exponential time exact algorithms for the SMC problem along with its hardness of approximation.

It is relevant to note that the paper Caragiannis et al. [36], does not give an algorithm to find a stable fractional matching that is non-integral, whenever one exists, irrespective of the social welfare. One of the contributions of our work is to fill this gap by designing a polynomial-time algorithm to find a stable fractional matching, which is non-integral, whenever one exists, under strict preferences.

In the simpler setting of binary values, the maximum weight matching satisfies many

[1] Note that a fractional matching is a convex combination of integral matchings.

[2] This further shows that fractional allocations in the stable matchings polytope studied in Roth et al. [103] and Teo and Sethuraman [114] need not in fact be stable themselves. In the follow-up work in Sethuraman et al. [109], the authors call the fractional allocations as fractional stable matchings, indicating that they are not discussing the stability of these fractional matchings but are only interested in their being a convex combination of stable matchings.

desirable properties such as being stable and giving no agent an incentive to misreport its preferences. This setting is studied by Bogomolnaia and Moulin [29]. However these results do not extend beyond binary values, not even to ternary values.

6.2 Notation and Preliminaries

6.2.1 Definitions

In this chapter, we consider fractionally matching n men to n women. We represent a fractional matching instance as $I = \langle M, W, \{u_i\}_{i\in[n]}, \{V_j\}_{j\in[n]}\rangle$. Here, $M = \{m_1, \cdots, m_n\}$ is the set of men and $W = \{w_1, \cdots, w_n\}$, is the set of women. The values of men and women are captured by valuation functions $u_1, \cdots u_n$ and $v_1, \cdots, v_n$ respectively. In particular, $u_i(j)$ is m_i's value for being matched integrally to w_j. Analogously, $v_j(i)$ is w_j's value for being matched integrally to m_i. We assume that all values are non-negative and that a strict linear order can be derived from the value of one agent. That is, for each man $m \in M$, there do not exist two distinct women w, w' such that $u_m(, w) = u_m(w')$. Similarly, for each woman $w \in W$, there do not exist two distinct men such that $v_w(m) = v_W(m')$.

Any cardinal valuation function will imply either a weak or a strict linear ordering over the set of agents. This ordering will be called the preference (relation) of this agent. When the value function implies a strict linear ordering, such preferences are called strict.

Fractional matchings may be viewed in two equivalent ways. The first is as a weight being assigned to each (m, w) pair such that the sum of the weight of any pair an agent is in is 1.

Definition 6.1 (Fractional Matching) *μ is said to be a fractional matching on $G = (M, W, E)$ if $\mu : m\times W \to [0, 1]$ such that for all $m \in M$, $\sum_{w\in W} \mu(m, w) \leq 1$ and for all $w \in W$, $\sum_{m\in W} \mu(m, w) \leq 1$.*

Fractional matchings can alternately be defined as convex combinations of integral matchings, where integral matchings are defined as the standard one-one matchings defined in Chapter 2. By the Birkhoff-von Neumann theorem [26], given a fractional matching as defined in 6.1, we can decompose it into a convex combination of $O(n^2)$ integral matchings. As a result, the only difference between the two viewpoints is that one considers a fractional matching independently, while the other looks at the integral matchings in its support as well. The support of a fractional matching is the set of all integral matchings whose convex combination forms the fractional matching (that is, with non-zero weight on the relevant edges). We will consider fractional matchings as defined in Definition 6.1.

We now present some notation with regard to fractional matchings. We shall say that fractional matching μ_1 is a subset of μ_2, when (i) they are both defined for the same instance I and (ii) for each $(m, w) \in M \times W$ such that if $\mu_1((m, w)) > 0$, then $\mu_1((m, w)) = \mu_2((m, w))$. That is, if a pair of agents is matched with some weight under μ_1, they must be matched with just a much weight under μ_2. μ_2 may match some agents who are entirely unmatched in μ_1, but the converse should not happen. Note that it is necessary for the underlying instance to be the same for this definition to make sense.

The value of a woman w under a fractional matching μ is $v_w(\mu) = \sum_{m \in M} \mu(m, w)V(m, w)$. Thus, it is essentially the weighted sum of the value from each of the integral matchings in the support of μ. The value of a man can be analogously defined as $u_m(\mu) = \sum_{w \in W} \mu(m, w)U(m, w)$. To define the notion of stability of fractional matchings, we first define the notion of a blocking pair in the context of fractional matchings.

Definition 6.2 (Blocking Pair for Fractional Matchings) *We say that (m, w) forms a* blocking pair *under matching μ if both get strictly less value from μ than they get by being matched* integrally *with each other.*

As in the rest of this thesis, a fractional matching μ is said to be stable is there does not exist a blocking pair of agents $(m, w) \in M \times W$ such that $U(m, w) > u_m(\mu)$ and $V(m, w) > v_w(\mu)$. The model studied in Gale and Shapley [52] looks at stable one-one matchings with ordinal preferences and show that stable integral matchings always exist. Clearly, we can always derive an instance of the type studied by Gale and Shapley given $I = \langle M, W, U, V \rangle$. Consequently, a stable integral matching always exists. Also, it is easy to see that integral matchings that are stable under our definition are also stable under the original definition of Gale and Shapley. We will, throughout our analysis, use these facts, without explicitly stating them.

Observe that to report their valuations, each agent need only report n values. Consequently, for any $j \in [n]$, the valuation function of w_j can be captured by a vector of length n V_j, where the i^{th} value in the vector $V_j[i] = v_j(i)$. Similarly for m_i, his valuations can be captured by an n length vector U_i. Our objective is to investigate the existence of incentive compatible mechanisms to find stable fractional matchings. In the context of this chapter, incentive compatibility can be defined as follows:

Definition 6.3 (Incentive Compatibility) *A matching mechanism that takes as input $U_1, \cdots, U_n$, $V_1, \cdots, V_n$ and returns a matching $\mu(U_1, \cdots, U_n, V_1, \cdots, V_n)$ is said to be incentive compatible if for each $w_j \in W$ and $m_i \in M$,*

$$v_j(\mu(V_j, , V_{-j}, U)) \geq v_j(\mu(x, V_{-j}, U))$$
$$\textit{and} \;\; u_i(\mu(U_i, U_{-i}, V)) \;\; \geq u_i(\mu(y, U_{-i}, V))$$

for any n-dimensional vectors of non-negative real values x and y.

We describe a special class of stable matching instances (which we call Conditional Mutual First Preference (CMFP)) having a *unique* stable fractional matching. In defining the CMFP class, we rely crucially on identifying pairs of nodes that satisfy a property called the mutual first preference (MFP) property.

Definition 6.4 (MFP) *We say a pair (m, w) satisfies mutual first preference property if $\{w\} = \text{argmax}_{a \in W} U(m, a)$, $\{m\} = \text{argmax}_{a \in M} V(a, w)$.*

That is, each happens to be the *first preference* of the other. Note that for any stable matching instance I with strict preferences, if there exists a pair of nodes that satisfies MFP, the two nodes must be matched under every stable matching.

An important contribution of our work is to give a polynomial-time algorithm to find a stable fractional matching, which is non-integral, whenever such a fractional matching exists. This algorithm is key to establishing that whenever there are no MFP pairs, under strict preferences, a stable fractional matching that is not integral can be found. This, in turn, characterizes the instances which have unique stable fractional matchings. envy graphs are critical to establishing these results. We now define envy and envy graphs for this context.

Definition 6.5 (Envy) *Given a stable matching instance $I = \langle M, W, U, V \rangle$ and a fractional matching μ, for any $w, w' \in W$, w is said to envy w' under μ if*

$$v_w(\mu) = \sum_{m \in M} \mu(m, w) V(m, w) < \sum_{m \in M} \mu(m, w') V(m, w).$$

Similarly, m is said to envy m' under μ if

$$u_m(\mu) = \sum_{w \in W} \mu(m, w) U(m, w) < \sum_{w \in W} \mu(m', w) U(m, w).$$

For an integral matching μ, w envies w' under μ if $v_w(\mu) = V(\mu(w), w) < V(\mu(w'), w)$. Given a matching, the envy graph under that matching as defined as follows.

Definition 6.6 (Envy Graph) *Given a stable matching instance $I = \langle M, W, U, V \rangle$ and a fractional matching μ, the envy graph on women under μ is a directed graph $G_W(\mu) = (W, E)$ where $(w, w') \in E \Leftrightarrow w$ envies w' under μ. Analogously, the envy graph on men under μ is a directed graph $G_M(\mu) = (M, E)$ where $(m, m') \in E \Leftrightarrow m$ envies m' under μ.*

When there are no MFP pairs, we can use envy graphs to find stable fractional matchings which are non-integral.

6.2.2 Overview of Main Results

We present an overview of all our main results before a detailed discussion and proofs in the next two sections.

Our first result shows that incentive compatibility and stability cannot coexist.

Lemma 6.1 *No mechanism which finds a stable fractional matching is incentive compatible, when the only condition on the values is that they are non-negative.*

We then extend this result to show that even an approximate notion of incentive compatibility cannot coexist with stability.

Theorem 6.1 *For $\epsilon \in [0, 1/2)$, there does not exist an ϵ-IC mechanism that always returns a stable fractional matching when the only condition on the values is that they are non-negative.*

Given the above two results, in Section 6.3.2 it is natural to try to identify a class of matching instances over which stability and incentive compatibility can be achieved together. To this end, we look for pairs of agents that must be matched in all stable matchings in a given instance.

Lemma 6.2 *Given any stable matching instance I , Algorithm 6.1 returns a matching that is a subset of any stable integral matching on I. Any stable fractional matching for instance I must set a weight of 1 on each pair contained in the matching returned by Algorithm 1.*

We show that whenever Algorithm 6.1 returns a perfect matching, truthful revelation of values is a Nash Equilibrium.

Theorem 6.2 *Given any matching instance belonging to Conditional Mutual First Preference ($CMFP$), any mechanism that finds a stable fractional matching is incentive compatible.*

The proof of this theorem leads to a corollary that contains a stronger result.

Corollary 6.1 *For each stable matching instance in $CMFP$, no coalition of agents can collude to strategically misreport their preferences (and improve their values under any mechanism) to find a stable fractional matching.*

We show that whenever an instance is not in $CMFP$, given any stable integral matching (for instance, the one returned by the Gale-Shapley algorithm), we can find a non-integral stable fractional matching by Algorithm 6.2. Consequently, we have the following result.

Theorem 6.3 *A stable matching instance I has a unique stable fractional matching if and only if it is in CMFP.*

Algorithm 6.2 defines an entire linear polytope of stable fractional matchings leading to the following corollary.

Corollary 6.2 *Under strict preferences, a stable matching instance has either a unique stable matching or uncountably many.*

6.2.3 Some Structural Observations

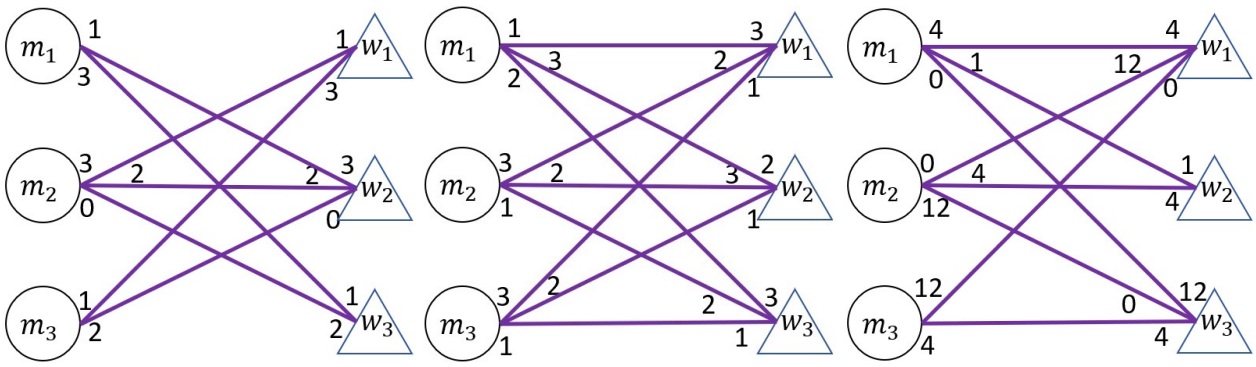
Before we present our analysis, it is important to demonstrate that it is non-trivial to compute stable fractional matchings which are not integral. We now make some structural observations regarding the space of stable fractional matchings.

Observation 6.1 *A convex combination of stable fractional matchings need not always be stable.*

Caragiannis et al. [36] illustrate this with an example which does not have strict preferences. Figure 6.1a demonstrates that this holds even with strict preferences. There are exactly two stable integral matchings. These are: $\mu_1 = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$ and $\mu_2 = \{(m_1, w_2), (m_2, w_1), (m_3, w_3)\}$. Fractional matching $\mu_\alpha = \alpha\mu_1 + (1-\alpha)\mu_2$ is not stable for all $\alpha \in [1/3, 2/3]$, as (m_3, w_2) form a blocking pair. Thus, we have that the space of stable fractional matchings is not easy to iterate over.

Note that for any instance, stable integral matchings form a subset of stable fractional matchings. However, knowing the space of stable integral matchings does not give much information about the space of stable fractional matchings. We shall now demonstrate this through two observations.

Observation 6.2 *A matching instance may have a unique stable integral matching but multiple stable fractional matchings.*



(a) Non-convexity (b) Multiple Stable Matchings (c) Unstable Support

Figure 6.1: Examples. Numeric values near the nodes are the corresponding values

Consider the stable matching instance represented in Figure 6.1b. The unique stable integral matching is $\mu_1 = \{(m_1, w_2), (m_2, w_1), (m_3, w_3)\}$. Consider matching $\mu_2 = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$. Fractional matching $\mu_\alpha = \alpha\mu_1 + (1-\alpha)\mu_2$ is stable for all $\alpha \in [1/2, 1]$.

This observation is consequential in conjunction with the observation that there may be stable fractional matchings with only unstable integral matchings in the support.

Observation 6.3 *There exist stable fractional matchings whose support consists solely of unstable integral matchings.*

This can be illustrated by the stable matching instance described in Figure 6.1c. The unique stable integral matching is $\mu_1 = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$. The matchings $\mu_2 = \{(m_1, w_2), (m_2, w_3), (m_3, w_1)\}$, $\mu_3 = \{(m_1, w_3), (m_2, w_1), (m_3, w_2)\}$ and $\mu_4 = \{(m_1, w_3), (m_2, w_2), (m_3, w_1)\}$ are all unstable. However, the fractional matching $\mu = \frac{\mu_2}{6} + \frac{\mu_3}{3} + \frac{\mu_4}{2}$ is in fact stable. Under μ, the values of the men are $u_1(\mu) = 1/6$, $u_2(\mu) = 4$ and $u_3(\mu) = 8$, while the values of the women are $v_1(\mu) = 4$, $v_2(\mu) = 13/6$ and $v_3(\mu) = 10$.

Observation 6.4 *Gender-optimal stable fractional matchings need not exist.*

It is well established that the Gale-Shapley algorithm matches each agent in the proposing side to their optimal stable partner. That is, agents on the proposing side are matched to their most preferred out of all agents they are matched to under any stable matching. Consequently, all agents on the proposing side get their maximum possible value under any stable matching. However, this does not extend to fractional matchings. Consider the matching instance described by Figure 6.1c. Here, under the only integral stable matching $\mu_1 = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$, all agents get a value of 4. This is the maximum possible

value for m_1 and w_2. Under the stable matching μ (defined in the previous observation), m_1 gets a value of $1/6$, w_2 gets a value of $13/6$, both of which are lower than their value under μ_1, while m_3 gets a value of 8 and w_3 gets a value of 10. As a result, both m_3 and w_3 are better off under μ, whereas m_1 and w_2 prefer μ_1.

These observations demonstrate that the space of stable fractional matchings is somewhat unintuitive, even under strict preferences which makes finding a stable non-integral, fractional matching non-obvious. Given any integral matching, it is not clear how to see whether or not it is present in the support of a stable fractional matching. Further, if so, what should be the weight on this matching. Our analysis on the incentive compatibility of stable fractional matching procedures provides a way to overcome these hurdles.

We first look at the coexistence of stability and incentive compatibility in a general setting, that is when the only condition on the values is that they are non-negative.

6.3 Co-existence of Stability and Incentive Compatibility

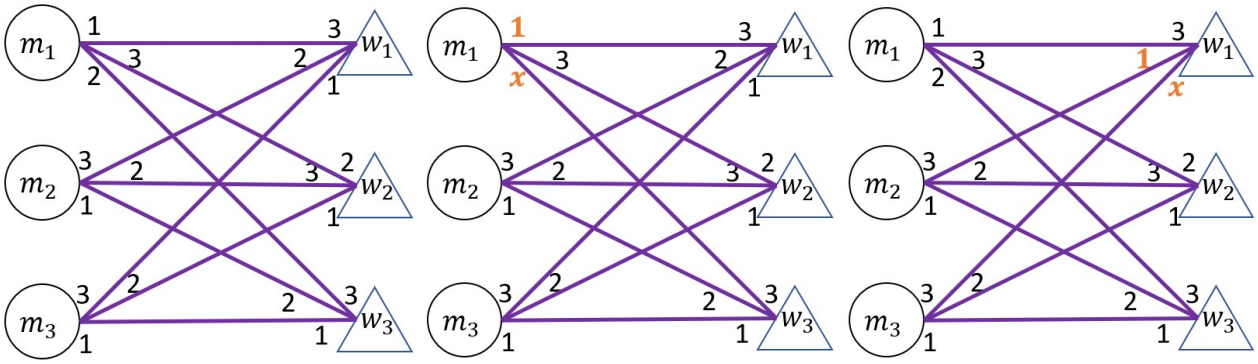
Incentive compatibility is very desirable for real-world matching applications. Roth [101] showed that there is no incentive compatible mechanism to find stable integral matchings. However, stable integral matchings form a proper subset of stable fractional matchings and it is not clear whether or not incentive compatible mechanisms exist in the fractional matching setting. We resolve this question by demonstrating that this, in fact, is not possible, even when we relax the notion of incentive compatibility to approximate incentive compatibility.

6.3.1 Impossibility in General Settings

We first assign cardinal values to the example used by Roth [101] to show that there is no incentive compatible mechanism to find a stable fractional matching. We then modify the example to show that in fact even with a relaxation in the notion of incentive compatibility, we have the same situation.

Lemma 6.1 *No mechanism which finds a stable fractional matching is incentive compatible, when the only condition on the values is that they are non-negative.*

Proof: Consider the stable matching instance described in Figure 6.2a. There are exactly two stable integral matchings $\mu_1 = \{(m_1, w_2), (m_2, w_1), (m_3, w_3)\}$ and $\mu_2 = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$. Further, the fractional matching $\mu_\alpha = \alpha\mu_1 + (1-\alpha)\mu_2$ is stable for all $\alpha \in [0, 1]$. No other fractional matching is stable for this instance. Thus, any stable fractional matching algorithm, when run on this instance, essentially chooses a value of $\alpha \in [0, 1]$ for μ_α.



(a) true values (b) m_1 misreports (c) w_1 misreports

Figure 6.2: A counterexample for incentive compatible mechanism

It is easy to see that m_1 will need to be matched to w_2 integrally, i.e. for the given stable fractional matching algorithm to choose value $\alpha = 1$. Let the algorithm choose $\alpha = c < 1$. m_1 can now misreport his preferences as shown in Figure 6.2b with $x \in [1, 3)$. Under this instance, the only stable fractional matchings are μ_α for $\alpha \in \left[\frac{x-1}{2}, 1\right]$. Thus, if a given algorithm chooses some $\alpha = c < 1$, there is an incentive for m_1 to misreport his preferences as shown in Figure 2b with $x > 2c + 1$, giving him strictly higher value. Similarly, w_1 would like to be matched integrally to m_1. Thus, she can misreport her preferences as in Figure 6.2c to ensure that she receives higher value whenever the algorithm chooses a value of $\alpha > 0$.

Thus no mechanism resulting in a stable fractional matching can be incentive compatible for all the agents when there are no restrictions on the input instances. □

We now look at relaxations of stability and incentive compatibility, which are defined as follows.

Definition 6.7 (ϵ-Stability) *A matching μ is said to be ϵ-stable, $\epsilon \in [0, 1)$, if for every $(m, w) \in M \times W$, we have that either $u_m(\mu) \geq (1 - \epsilon)U(m, w)$ or $u_w(\mu) \geq (1 - \epsilon)V(m, w)$.*

Definition 6.8 (ϵ-IC) *A matching mechanism that returns a matching $\mu(U_1, \cdots, U_n, V_1, \cdots, V_n)$ is said to be ϵ-IC, $\epsilon \in [0, 1)$, if for each $w_j \in W$ and $m_i \in M$,*

$$\begin{aligned} & v_j(\mu(V_j, , V_{-j}, U)) \geq (1 - \epsilon)v_j(\mu(x, V_{-j}, U)) \\ \textit{and} \quad & u_i(\mu(U_i, U_{-i}, V)) \geq (1 - \epsilon)u_i(\mu(y, U_{-i}, V)) \end{aligned}$$

for any n length vectors of non-negative values x and y.

We shall first examine the existence of an ϵ-incentive compatible mechanism for finding a stable fractional matching.

Theorem 6.1 *For $\epsilon \in [0, 1/2)$, there does not exist an ϵ-IC mechanism that always returns a stable fractional matching when the only condition on the values is that they are non-negative.*

Proof: Consider the matching instance shown in Figure 6.3a. This is the same as the instance in Figure 6.2a, with every value of 3 replaced by $k \geq 3$. Analogous to the proof of Lemma 6.1, we have that whenever the value of m_1 or w_1 is less than $(1-\epsilon)k$, any mechanism that finds a stable matching will not be $\epsilon-$IC and will give the agents an incentive to misreport their values as shown in Figures 6.3b and 6.3c respectively.

Recall from the proof of Lemma 6.1, the only stable integral matchings are, $\mu_1 = \{(m_1, w_2), (m_2, w_1),$ and $\mu_2 = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$. Further, any stable fractional matching mechanism must return a matching $\mu_\alpha = \alpha\mu_1 + (1-\alpha)\mu_2$. Note that $u_{m_1}(\mu_\alpha) = 2 + (k-2)\alpha$ and $u_{w_1}(\mu_\alpha) = k - (k-2)\alpha$.

For any mechanism to be ϵ-IC, it must select a matching μ_α such that for w_1: $k-(k-2)\alpha \geq (1-\epsilon)k$. This implies that we need

$$\frac{k}{k-2}\epsilon \geq \alpha.$$

Similarly for m_1, $2 + (k-2)\alpha \geq (1-\epsilon)k$. Consequently, to satisfy ϵ-IC for m_1, we need that

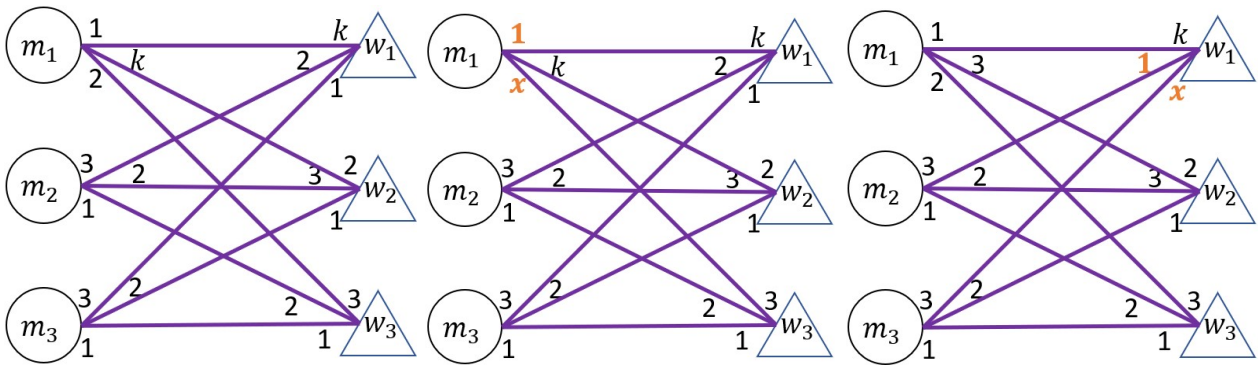
$$\frac{k}{k-2}\epsilon \geq (1-\alpha).$$

Therefore we require

$$\max(\alpha, 1-\alpha) \leq \frac{k}{k-2}\epsilon.$$

Now the LHS of this condition is minimized for $\alpha = 1/2$. It is easy to see that for any $\epsilon \in [0, 1/2)$, there exists a large enough real valued $k > 2$ such that $\frac{k}{k-2}\epsilon < 1/2$. Thus for any $\epsilon \in [0, 1/2)$, there cannot exist an ϵ-IC mechanism that always returns a stable fractional matching when the only condition on the values is that they are non-negative. This proves the result. □

Before moving to achieving exact incentive compatibility, we shall first describe a $1/2$-IC mechanism which produces a $1/2$-stable fractional matching. The mechanism essentially runs the Gale-Shapley algorithm with the men as the proposing side once, getting μ_M and the women as the proposing side once, getting μ_W. It then returns a matching which assigns a weight equal to half on each of the matchings obtained $1/2(\mu_M + \mu_W)$. From Roth [101], the Gale-Shapley algorithm is IC for the proposing side and as a result, the described mechanism is $1/2-$IC. The matching is also $1/2$-stable by a similar argument. In fact, for this mechanism,



(a) true values (b) m_1 misreports (c) w_1 misreports

Figure 6.3: Counterexample for $\epsilon-$IC mechanisms, $\epsilon \in [0, 1/2)$

these bounds are tight. It would be interesting to see if any other mechanism can improve upon these bounds.

6.3.2 Incentive Compatibility under Restricted Settings

We know from Roth [101] that there is no incentive compatible mechanism for finding stable integral matchings in general. However, when there is a unique stable integral matching, we have an exception. For a mechanism implementing the Gale-Shapley algorithm, truthful revelation of preferences by all agents forms a Nash equilibrium as a simple consequence of prior work. Roth [101] showed that when the Gale-Shapley algorithm is run with men as proposers, it is a dominant strategy for men to be honest. As shown by Teo et al. [115], women can find their optimal manipulation for a given instance in polynomial time. This optimal manipulation matches her with her best possible partner under the Gale-Shapley algorithm, when men propose. It is shown by Vaish and Garg [119] that the resultant matching is also stable under the true preferences. Thus, each woman's partner, even after the optimal manipulation, will remain the same.

Instances with Unique Stable Fractional Matchings

This motivates us to explore a class of instances where there is a unique stable fractional matching and look for an incentive compatible mechanism for this class. While there has been much work on finding stable integral matchings, so far there is no work which aims at finding a stable matching which is non-integral. For some instances, such a matching need not exist. Previous work has established that there is always a stable integral matching and therefore one degenerate fractional matching always exists. It will be nice to have an algorithm, which, when given a stable matching instance, finds a stable fractional matching

that is non-integral whenever one exists. To this end, we first try to categorize the class of instances where there exists a unique stable fractional matching. Note that there is a *unique* stable fractional matching in the following settings.

1. An idealistic setting where if m's first preference is w, then w's first preference is m. Recall that we call such pairs of nodes as MFP pairs (mutual first preference). Here it is easy to see that the unique stable matching is the one where everyone is matched to their first preference.

2. All men having identical preferences and all women have identical preferences. In chapter 4, we call such a setting as one with ranked values, albeit in the context of many-to-one matchings. Here, the only stable matching is where the i^{th} ranked man is matched with the i^{th} ranked woman for all $i \in [n]$. This is because the highest ranked man and woman must be matched as they are each other's first preference. Given this, now the second highest ranked man and woman become each other's first preference as the most popular man and most popular women are now unavailable. In fact any combination of these two settings will have a unique stable fractional matching.

Note that if any stable matching instance has MFP pairs, any stable matching must match them. Based on this, we provide the following polynomial-time algorithm which returns a matching that must be a subset of any stable matching.

Algorithm 6.1: Mutual first preference matching algorithm

Input: $I = \langle M, W, U, V \rangle$
Output: $\mu, I' = \langle M', W', U', V' \rangle$

$t \leftarrow 0$, $\mu \leftarrow \emptyset$;
$M^{(0)} \leftarrow M$, $W^{(0)} \leftarrow W$, $U^{(0)} \leftarrow U$, $V^{(0)} \leftarrow V$;
$I^{(0)} = \langle M^{(0)}, W^{(0)}, U^{(0)}, V^{(0)} \rangle$;
while $\exists (m, w)$ *s.t.* $w = \text{argmax}_{a \in W^{(t)}} U^{(t)}(m, a)$ *AND* $m = \text{argmax}_{a \in M^{(t)}} V^{(t)}(a, w)$ **do**
    $\mu \leftarrow \mu \cup (m, w)$;
    $I^{(t+1)}$ is $I^{(t)}$ with m and w removed;
    $t \leftarrow t + 1$;
$I' \leftarrow I^{(t)}$;

We use the above algorithm to define a special class of stable matching instances called **Conditional Mutual First Preference (CMFP)**.

Definition 6.9 *We say that a stable matching instance I is in class CMFP if Algorithm 1 returns a perfect matching when I is given as input.*

Lemma 6.2 *Given any stable matching instance I , Algorithm 6.1 returns a matching that is a subset of any stable integral matching on I. Any stable fractional matching for instance I must set a weight of 1 on each pair contained in the matching returned by Algorithm 1.*

Proof: We prove this result by contradiction. Assume the result is not true. That is, there is a stable matching instance I where μ is the matching returned by Algorithm 1 on I, and there is a stable matching μ' on I such that μ is not a subset of μ'. Let$|\mu| = k > 0$ (If $|\mu| = 0$ then μ is the empty matching, and as a result is a subset of every matching on I.). Let $i_1, \cdots, i_k$ and $j_1, \cdots, j_k$ such that $\mu^* = \{(m_{i_1}, w_{j_1}), \cdots, (m_{i_k}, w_{j_k})\}$ where (m_{i_t}, w_{j_t}) are matched in the t^{th} round in Algorithm 1.

Let $t^* \in [k]$ be the lowest index in $[k]$ such that $\mu'(m_{i_{t^*}}, w_{j_{t^*}}) < 1$. As μ' is stable, at least one of $m_{i_{t^*}}$ and $w_{j_{t^*}}$ must have higher value for μ' than from matching integrally with each other. Without loss of generality, let this be $m_{i_{t^*}}$. By construction of Algorithm 1, this is only possible when $\mu'(m_{i_{t^*}}, w_{j_{t'}}) > 0$ for some $t' < t^*$. Thus, $\mu'(m_{i_{t'}}, w_{j_{t'}}) < 1$. This is a contradiction as we assumed t^* to be the lowest index for which this happens. This proves the result. □ As a result, we have a polynomial-time algorithm to determine when an instance is CMFP. Algorithm 1 helps us identify certain edges which must be included in any stable matching. Further, if the matching returned is a perfect matching, then it is the unique stable fractional matching for the given instance. This is a consequence of Lemma 6.2. The class CMFP is also special in that incentive compatibility can be achieved for this class.

Incentive Compatibility under CMFP Instances

Theorem 6.2 *Given any matching instance belonging to Conditional Mutual First Preference ($CMFP$), any mechanism that finds a stable fractional matching is incentive compatible.*

Proof: Given a stable matching instance $I \in CMFP$, every mechanism generating a stable fractional matching will return the same matching μ^*. We use Algorithm 1 to give us labellings $i_1, \cdots, i_n$ and $j_1, \cdots, j_n$ such that $\mu^* = \{(m_{i_1}, w_{j_1}), \cdots, (m_{i_n}, w_{j_n})\}$ where (m_{i_t}, w_{j_t}) are matched in the t^{th} round in Algorithm 1.

Let all other agents be truthful. Clearly, (m_{i_1}, w_{j_1}) have no incentive to misreport their preferences as they are already matched to their first preference. As long as m_{i_1} and w_{j_1} stay truthful, they will continue to be matched to each other, irrespective of how other agents are behaving. Now for $t > 1$ for (m_{i_t}, w_{j_t}) neither can gain by increasing or decreasing their value for any agent matched earlier. This is because the agents who are matched before round t are truthful and will not become MFP pairs with m_{i_t} or w_{j_t}. Thus, those pairings will not change. Of the remaining agents, m_{i_t} and w_{j_t} have highest value for each other and cannot benefit

from misreporting their preferences. Consequently, when all other agents are truthful, no agent has an incentive to misreport its preferences. □

In fact, the same reasoning also implies that the stable matching in a CMFP instance is also robust to coalitional manipulation. Note that the nodes matched to their first preferences have no incentive to deviate. The remaining are unable to misreport so that they would be matched to a node which has already been matched earlier by Algorithm 1. These nodes have no incentive to misreport so as to match with nodes matched later. As a result, there is no way for a coalition of agents to misreport preferences and increase their values.

Corollary 6.1 *For each stable matching instance in $CMFP$, no coalition of agents can collude to strategically misreport their preferences (and improve their values under any mechanism) to find a stable fractional matching.*

Consequently, when the matching instances are restricted to those in $CMFP$, we have incentive compatible mechanisms that produce stable fractional matchings. What we now show is that in fact, this is the only set of matching instances where there is a unique stable fractional matching.

6.4 Characterizing Instances with Unique Stable Matchings

We now design a polynomial-time algorithm to find a stable fractional matching which is non-integral, whenever $I \notin CMFP$, starting from a stable integral matching (By Gale and Shapley [52], such a matching always exists and can be found efficiently.). To the best of our knowledge, there is no algorithm in prior work to find a stable fractional matching that is non-integral. When the preferences are strict, we utilize envy graphs to help find stable fractional matchings given a stable integral matching. Note that by Lemma 6.2, the matching returned by Algorithm 1 must be a subset of any stable matching. As a result, it suffices to have an algorithm that works on instances without MFP pairs. We can first run Algorithm 1 and then use it on the reduced instance returned by Algorithm 1.

The key idea is as follows. Under strict preferences, when there are MFP pairs, it is not possible to reduce the weight on the edge between them and still be stable. However, whenever there are no MFP pairs, each edge matched under a stable matching will have at least one node who prefers another agent to their current partner. To this end, we use envy graphs to construct other matchings which improve the value of some of the agents, without creating blocking pairs. As the preferences are strict, there will always exist a small enough weight to place on these matchings such that there are no blocking pairs with other nodes. While it is not necessary to explicitly construct envy graphs to help find such matchings,

envy graphs help in expressing these ideas in an intuitive manner. While this work presents the first exploration of the intersection of stability and incentive compatibility in fractional matchings, there is still much to be studied in this area. It would be interesting to see if relaxing stability can help achieve incentive compatibility or whether there are other classes of matching instances for which incentive compatibility can be achieved.

6.4.1 Envy Graphs

In this section, we introduce cycles in envy graphs and discuss acyclic envy graphs.

Cycles in Envy Graphs

Let μ_1 be a stable integral matching and assume $G_W(\mu_1)$ contains a cycle. We can produce an alternate matching μ_2 where all women not in C are matched as in μ_1 and each woman in C is matched to her successor's partner under μ_1. The successor of a node on a directed path or a cycle is the vertex to which it has an outgoing edge in the path/cycle. There can only be one such vertex.This resultant matching need not be stable, even if the original matching was stable. However, it will increase the value of the women in the cycle. Define $\mu_\alpha = \alpha\mu_1 + (1-\alpha)\mu_2$, $\alpha \in [x, 1]$.

Let us now discuss the stability of μ_α. In this case, no woman sees a decrease in value. As a result, it suffices to ensure that no man experiences a large enough drop in value for a blocking pair to form. Let us first consider men whose partners are unchanged. They do not form any blocking pairs. For these men, any woman they may prefer to their current partner has the same or higher value than in the original matching. Now, consider the men whose partners do change. Note that as μ_1 is stable, these men prefer their partners under μ_1 over those under μ_2, otherwise μ_1 would not be stable. Let us define $W_m(\mu) = \{w \in W : V(m, w) > v_w(\mu)\}$. In order for μ_α, $\alpha \in [x, 1]$ to be stable we need that for each $m \in M$ such that $\mu_1(m) \neq \mu_2(m)$, $xu_m(\mu_1) + (1-x)u_m(\mu_2) \geq \max_{w \in W_m(\mu_1)} U(m, w)$. As we have strict preferences, a value of $x < 1$ can be found in polynomial-time. Analogously, a fractional matching can also be found when there is a cycle in $G_M(\mu)$.

Acyclic Envy Graphs

In the absence of a cycle and any MFP pairs, a path must necessarily exist in at least one of $G_M(\mu_1)$ or $G_W(\mu_1)$. Let this be path P in $G_W(\mu_1)$. Now we can create an alternate matching μ_2 as before where each woman on the path is matched to the partner of her successor, with the sink being matched to the partner of the source. This alone will not be enough to make a stable matching. This is because the sink of P, say w, and her partner, say m, will both get value less than that in μ_1, and form a blocking pair. In order to mitigate this, m needs

an increase in value (w, being a sink, is matched to her favourite man, so she cannot see an increase in value by matching with anyone else). To this end, we find a path in $G_M(\mu_1)$ starting from m. This must exist, as there are no MFP pairs and w is a sink. As before, this path is used to generate an alternate matching. In this manner, we can continue till there are no blocking pairs possible.

These matchings form the support of the stable fractional matching. To achieve stability, we need to set appropriate weights on the matchings. This is possible as the preferences are strict. As in the case of when cycles are present, a linear program can be solved to find the weights on μ_1 and all newly defined matchings to find a stable fractional matching.

6.4.2 Algorithm for Generating Stable Non-Integral Matchings

Before detailing the algorithm, we first set some notation. $G_M(\mu)$ and $G_W(\mu)$ are the envy graphs of men and women under μ as defined. For a path P, $sink(P)$ denotes the last node in the directed path. The $UpdateCycle$ routine takes as input a matching μ and a cycle in either $G_M(\mu)$ or $G_W(\mu)$ and matches each agent in the cycle to their successor's (the agent to whom they have an outgoing edge in the cycle) partner. All agents whose matching is not defined by this are matched as in μ. UpdatePath works almost identically, with the sink of the path being matched to the source's partner.

Correctness of Algorithm 2

The correctness of the Algorithm hinges on being able to correctly find the required values for $x_1, \cdots, x_k$. Observe the coefficients of the linear program described. By Farkas' Lemma, this must be feasible, whenever preferences are strict. As a result, we have that Algorithm 2 correctly finds a stable non-integral matching, whenever the preferences are strict and the instance is not in CMFP.

Time Complexity of Algorithm 2

A cycle or a path in an envy graph can always be found in polynomial-time. Thus, computing a new matching can be done in polynomial-time. As each matching is defined to improve the value of a previously unimproved agent, there can be at most $2n$ matchings. Further, the required x_i values can be found by solving a linear program with the appropriate constraints. The number of variables is k, which is the number of matchings. Thus, this linear program can be solved efficiently. Consequently, the algorithm detailed is a polynomial-time algorithm. It finds a stable fractional matching which is not integral whenever one exists.

Theorem 6.3 *A stable matching instance I has a unique stable fractional matching if and only if it is in CMFP.*

Algorithm 6.2: Algorithm for generating stable non-integral matchings

Input: Instance $I = \langle M, W, U, V \rangle$, stable matching μ_1
Output: μ

$(\mu_s, I') \leftarrow$ CMFP_matching(I);
$\mu \leftarrow \mu_s$;
if *I' is non-empty* **then**
  $k \leftarrow 1$;
  Generate Envy Graphs $G_M(\mu_1)$ and $G_W(\mu_1)$;
  if *Either $G_M(\mu_1)$ or $G_W(\mu_1)$ contain a cycle C* **then**
    $k \leftarrow 2$;
    $\mu_2 \leftarrow UpdateCycle(\mu_1, C)$;
    $A \leftarrow C$
  else
    $A \leftarrow \emptyset$ \\ Set of agents whose values increase;
    Find sink node $w_1 \in G_W(\mu_1)$;
    $currNode \leftarrow \mu_1(w_1)$;
    $currSet \leftarrow M$;
    while $currNode \notin A$ **do**
      $k \leftarrow k + 1$;
      Find path P in $G_{currSet}(\mu_1)$ starting from $currNode$;
      $\mu_k \leftarrow UpdatePath(\mu_1, P)$;
      $A \leftarrow A \cup (P \setminus \{sink(P)\})$;
      $currNode \leftarrow \mu_1(sink(P))$;
      if $currSet = M$ **then**
        $currSet \leftarrow W$;
      else
        $currSet \leftarrow M$;
  Find $x_1, \cdots x_k \in [0, 1)$ such that $\sum_{i=1}^{k} x_i = 1$ and
  for all $m \in M \cap A$, $\sum_{i=1}^{k} x_i U(m, \mu_i(m)) \geq u_m(\mu_1)$,
  for all $w \in W \cap A$, $\sum_{i=1}^{k} x_i V(\mu_i(w), w) \geq v_w(\mu_1)$,
  for all $m \in M \setminus A$, $\sum_{i=1}^{k} x_i U(m, \mu_i(m)) \geq \max_{w \in W : U(m,w) < u_m(\mu_1)} U(m, w)$ and
  for all$w \in W \setminus A$, $\sum_{i=1}^{k} x_i V(\mu_i(w), w) \geq \max_{m \in M : V(m,w) < v_w(\mu_1)} V(m, w)$
  $\mu \leftarrow \mu_s \cup \sum_{i=1}^{k} x_i \mu_i$;

Proof: Let I be a stable matching instance. We have already established that if $I \in CMFP$, I has a unique stable fractional matching. We now prove the other direction by establishing the contrapositive. Let $I \notin CMFP$, and μ_s and I' be the matching and instance returned by Algorithm 1 on I. Clearly, μ_s is not a perfect matching. Any matching μ on I' which is stable, can be combined with μ_s to obtain a stable matching for I.

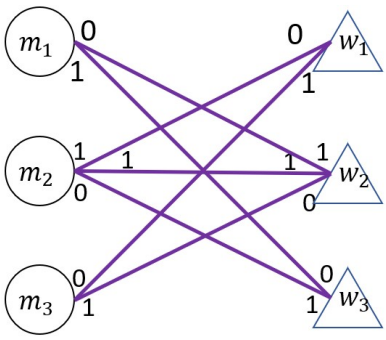


Figure 6.4: Algorithm 6.2 fails when there are ties

Let μ_1 be the stable matching returned by the Gale-Shapley algorithm on I' with men proposing. Note that, as I' is the instance returned by Algorithm 1, there is are no MFP pairs. Consequently, at least one of $G_M(\mu_1)$ and $G_W(\mu_1)$ will be non-empty. Algorithm 2 will give a stable fractional matching which is not integral. □ A simple corollary of Theorem 6.3 is that under strict preferences, a stable matching instance has either a unique stable fractional matching, or uncountably many.

Corollary 6.2 *Under strict preferences, a stable matching instance has either a unique stable matching or uncountably many.*

6.4.3 Weak Preferences

In the above algorithm, we have assumed that the preferences of the agents on either side are strict. That is, the values are such that no two agents on the opposite side are equal. Hence, the value of each agent implies a linear order over the agents on the other side. Note that, this assumption is rather common and not particularly restrictive. In fact, in the absence of this assumption, our results on CMFP continue to hold. However, Algorithm 6.2 requires strict preferences to be execute correctly. To see why, consider the matching instance described in Figure 6.4.

In this example, one stable matching is $\mu_1 = \{(m_1, w_3), (m_2, w_1), (m_3, w_2)\}$. This is a men-optimal stable matching where each man gets value 1 and each woman gets value 0. There is an envy cycle where w_1 envies w_2, w_2 envies w_3 and w_3 envies w_1 in turn. In this case, Algorithm 6.2 will try to find a convex combination of $\mu_2 = \{(m_1, w_2), (m_2, w_3), (m_3, w_1)\}$ and μ_1. However, any convex combination that is not μ_1 or μ_2 will not be stable as it will give both m_2 and w_2 value less than 1, making (m_2, w_2) a blocking pair.

6.5 Conclusions and Future Work

We initiated work on the existence of incentive compatible mechanisms that find stable fractional matchings. We showed that without any additional assumptions, no such mechanism exists. We then characterized the space of matching instances with unique stable fractional matchings. When restricted to this space, all stable matching mechanisms are incentive compatible. An interesting direction of future work is to identify if other conditions that allow for incentive compatibility. A more detailed discussion is in Chapter 7.

Chapter 7

Summary and Future Work

In this thesis, we studied the problem of algorithmically achieving fairness and efficiency in different matching settings. We first looked at settings with one-sided preferences, where fairness was defined as being almost envy-free and efficiency was studied through the lens of maximizing social welfare. We then looked at achieving fairness and efficiency in the setting of matchings with preferences on both sides. Stability is a key efficiency requirement in such settings. Consequently, we looked at fairness for stable matchings. We now summarize the contributions and future work for each of the technical chapters.

7.1 Summary of our Contributions

Part 1: One-Sided Preferences

Chapter 3: Fair and Efficient Matching of Delivery Tasks to Agents

This chapter introduced a novel problem of fair distribution of delivery orders on tree graphs. Focusing on tree graphs, we established a comprehensive characterization of the space of instances that admit fair (EF1 or MMS) and efficient (SO or PO) allocations. We analyzed the complexity of finding such allocations (or deciding if they exist) and while we identified this problem as computationally hard, we developed an XP algorithm parameterized by the number of agents for each combination of fairness-efficiency notions. Complementing our theoretical findings, we conducted several experiments on randomly generated graphs.

Chapter 4: Repeatedly Matching Items to Agents Fairly and Efficiently

This chapter initiated the study of repeated matchings with *history-dependent* valuations. We explored efficiency through the lens of social welfare maximization and fairness through a

variety of relaxations of envy-freeness. We found that even in settings where social welfare can be maximized and EF1 repeated matchings can be found tractably, maximizing social welfare over EF1 matchings is APX-Hard. We found that pre-existing relaxations of envy-freeness need not always exist for simple repeated matching settings and propose the notion of swapEF to resolve this.

Part 2: Two-Sided Preferences

Chapter 5: Achieving Fairness and Stability in Many-to-one Matchings

This chapter initiated the study of finding a fair and stable many-to-one matching under cardinal valuations. We studied leximin optimality, which has been hitherto unexplored for stable matchings. We gave efficient algorithms to find the leximin optimal stable matching under rankings with strict preferences. In the absence of rankings, but with strict preferences, we gave an algorithm to find the leximin optimal stable matching when the number of colleges is fixed to two. We then showed that when $n = \Omega(m)$, then, under strict preferences without rankings, the problem becomes intractable. We showed that even with weak rankings, finding a leximin optimal stable matching is intractable. In fact, it is NP-Hard to find even a polynomial approximation with general valuations.

Chapter 6: Stability and Incentive Compatibility in Fractional Matchings

This chapter has looked into the design of incentive compatible mechanisms for finding stable fractional matchings. We first showed that this is not possible under unrestricted cardinal utilities, even when we relax the the notion of incentive compatibility up to a half-approximation. We then discovered a class of stable matching instances which have a unique stable fractional matching, namely those satisfying the conditional mutual first preference (CMFP) property. We showed that *every* mechanism that finds a stable fractional matching is incentive compatible if and only if the input instances are in CMFP. We presented an algorithm (Algorithm 2) that computes, under strict preferences, stable fractional matchings which are non-integral (to the best of our knowledge, this is the first such algorithm). This algorithm makes intelligent use of envy-graphs, hitherto unused in the stable matchings literature.

7.2 Future Directions of Research

We assert that we have settled a few important problems in the topic of co-existence of fairness and efficiency in fractional matchings. Clearly this does not in anyway settle all the issues in this important topic. Our work suggests several interesting directions of future work.

Part 1: One-Sided Preferences

In general, for the first part, we introduce two new models for fair division. For both settings we show hardness results, and it would be interesting to see if approximation schemes can be designed. Further, in both of these models, more generality possible. For instance, for the case of fair distribution of delivery tasks (Chapter 3), more general graph structures need to be considered. For repeated matchings (Chapter 4), this manifests as resolving the existence of EF1 or swapEF repeated matchings.

Further, introducing heterogeneity in the agent valuations would be another interesting direction. For the delivery setting, this could be done with differing feasibility constraints for agents, different edge costs, or even adding a service cost of servicing the vertices. For repeated matchings, the agent values for their bundles need not be additive, but may be submodular or even subadditive [11, 18, 19]. Parallel to these, a natural question would be what other fairness notions would be appropriate in these spaces. There are many other fairness concepts prevalent like Equitability [48], Maximum Nash Welfare [11, 17, 19, 35] and even the more general P-mean welfare [10, 39]. Finally, another interesting path would be theoretically bounding the price of fairness on efficiency for either setting.

Part 2: Two-Sided Preferences

In general, it is important to identify fairness notions that maintain fairness across either side for stable matchings. We focus on leximin, but it is possible that other approaches may also work. For example, Karni et al. [71] use PIIF, first defined by Kim et al. [72], for many-to-one stable matchings. However, they maintain fairness for only one-side. Here, a price opf stability can also be studied, to understand if stability imposes a loss in fairness.

Specifically for many-to-one stable matchings (Chapter 5) one open problem is whether for more than two colleges, with strict preferences, an exact algorithm may be possible. Another potential direction may be finding an a more general subclass of matching instances where it is possible to find a fair or approximately-fair and stable matching. While our results have ruled out approximations for additive valuations in general, under isometric valuations or general strict preferences, approximations may still be possible.

For stable fractional matchings, the hardness of finding a beneficial manipulation for our algorithm is not clear. Another relevant direction of future work would be to find algorithms that produce stable fractional matchings and are hard to manipulate. One more possibility would be to investigate if it is possible to achieve incentive compatibility by relaxing the stability constraint. An interesting problem is whether the use of envy-graphs can yield better

approximation algorithms for finding social welfare maximizing stable fractional matchings. Our analysis hinges on the fact that there will always be envy whenever the instance does not have any MFP pairs. As a result, an interesting future direction would be to look at envy-free matchings and try and see if there are any incentive compatible mechanisms to find envy-free matchings.

Going ahead, it would be interesting to explore these research directions. We believe that our work will encourage further research on fair and efficient matchings.

Bibliography

[1] Atila Abdulkadiroğlu and Tayfun Sönmez. School choice: A mechanism design approach. *American Economic Review*, 93(3):729–747, 2003. 2

[2] Atila Abdulkadiroğlu, Parag A Pathak, and Alvin E Roth. Strategy-proofness versus efficiency in matching with indifferences: Redesigning the NYC high school match. *American Economic Review*, 99(5):1954–78, 2009. 8

[3] Martin Aleksandrov. Envy freeness up to one item: Shall we duplicate or remove resources? *Progress in Artificial Intelligence: 21st EPIA Conference on Artificial Intelligence (EPIA 2022)*, pages 727–738, 2022. 59

[4] Georgios Amanatidis, Evangelos Markakis, Afshin Nikzad, and Amin Saberi. Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms (TALG)*, 13(4):1–28, 2017. 20, 87

[5] Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A Voudouris, and Xiaowei Wu. Fair division of indivisible goods: A survey. *arXiv preprint arXiv:2208.08782*, 2022. 6, 18

[6] Haris Aziz and Bettina Klaus. Random matching under priorities: stability and no envy concepts. *Social Choice and Welfare*, 53(2):213–259, 2019. 8, 21

[7] Haris Aziz, Serge Gaspers, Zhaohong Sun, and Toby Walsh. From matching with diversity constraints to matching with regional quotas. *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2019)*, pages 377–385, 2019. 22

[8] Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi, and Toby Walsh. Fair allocation of indivisible goods and chores. *Autonomous Agents & Multi Agent Systems*, 36(3):1–21, 2022. 18, 19, 28, 74

[9] Haris Aziz, Bo Li, Herve Moulin, and Xiaowei Wu. Algorithmic fair allocation of indivisible items: A survey and new questions. *ACM SIGecom Exchanges*, 20(1):24–40, 2022. 6, 18

[10] Siddharth Barman and Sanath Kumar Krishnamurthy. Approximation algorithms for maximin fair division. *ACM Transactions on Economics and Computation (TEAC)*, 8(1): 1–28, 2020. 2, 8, 19, 20, 28, 32, 87, 144

[11] Siddharth Barman and Ranjani G. Sundaram. Uniform welfare guarantees under identical subadditive valuations. *Twenty-Ninth International Joint Conference on Artificial Intelligence, (IJCAI 2020)*, pages 46–52, 2020. 20, 59, 87, 144

[12] Siddharth Barman and Paritosh Verma. Existence and computation of maximin fair allocations under matroid-rank valuations. *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2021)*, pages 169–177, 2021. 8, 28

[13] Siddharth Barman and Paritosh Verma. Truthful and fair mechanisms for matroid-rank valuations. *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI 2022)*, 36:4801–4808, 2022. 28

[14] Siddharth Barman, Arpita Biswas, Sanath Kumar Krishnamurthy, and Yadati Narahari. Groupwise maximin fair allocation of indivisible goods. *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI 2017)*, pages 9921–9922, 2017. 8, 19, 59

[15] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding fair and efficient allocations. *Proceedings of the 19th ACM Conference on Economics and Computation (EC 2018)*, pages 557–574, 2018. 8, 18, 31

[16] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Greedy algorithms for maximizing nash social welfare. *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*, pages 7–13, 2018. 20, 59, 87

[17] Siddharth Barman, Ganesh Ghalme, Shweta Jain, Pooja Kulkarni, and Shivika Narang. Fair division of indivisible goods among strategic agents. *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2019)*, pages 1811–1813, 2019. 8, 18, 19, 20, 31, 72, 144

[18] Siddharth Barman, Umang Bhaskar, and Nisarg Shah. Optimal bounds on the price of fairness for indivisible goods. *Proceedings of the 16th International Conference on Web and Internet Economics (WINE 2020)*, pages 356–369, 2020. 8, 144

[19] Siddharth Barman, Anand Krishna, Pooja Kulkarni, and Shivika Narang. Sublinear approximation algorithm for Nash social welfare with XOS valuations. *arXiv preprint arXiv:2110.00767*, 2021. 8, 18, 144

[20] Surender Baswana, Partha Pratim Chakrabarti, Sharat Chandran, Yashodhan Kanoria, and Utkarsh Patange. Centralized admissions for engineering colleges in India. *Proceedings of the 20th ACM Conference on Economics and Computation (EC 2019)*, pages 323–324, 2019. 86

[21] Nawal Benabbou, Mithun Chakraborty, Ayumi Igarashi, and Yair Zick. Finding fair and efficient allocations when valuations don't add up. *Proceedings of the 13th Symposium on Algorithmic Game Theory (SAGT 2020)*, pages 32–46, 2020. 8, 21, 28

[22] Kristóf Bérczi, Erika R. Bérczi-Kovács, Endre Boros, Fekadu Tolessa Gedefa, Naoyuki Kamiyama, Telikepalli Kavitha, Yusuke Kobayashi, and Kazuhisa Makino. Envy-free relaxations for goods, chores, and mixed items. *arXiv preprint arXiv:2006.04428*, 2020. 18

[23] Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *ACM SIGecom Exchanges*, 5(3):11–18, 2005. 20, 88, 92

[24] Umang Bhaskar, A. R. Sricharan, and Rohit Vaish. On approximate envy-freeness for indivisible chores and mixed resources. *Proceedings of the 25th International Conference on Randomization and Computation and the 24th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX/RANDOM 2021)*, pages 1:1–1:23, 2021. 18, 28

[25] Vittorio Bilò, Ioannis Caragiannis, Michele Flammini, Ayumi Igarashi, Gianpiero Monaco, Dominik Peters, Cosimo Vinci, and William S Zwicker. Almost envy-free allocations with connected bundles. *Games and Economic Behavior*, 131:197–221, 2022. 28

[26] Garrett Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucuman, Ser. A*, 5:147–154, 1946. 124

[27] Arpita Biswas and Siddharth Barman. Fair division under cardinality constraints. *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 91–97, 2018. 19, 56, 58, 66, 67, 72

[28] Shantanu Biswas and Y Narahari. Object oriented modeling and decision support for supply chains. *European Journal of Operational Research*, 153(3):704–726, 2004. 29

[29] Anna Bogomolnaia and Hervé Moulin. Random matching under dichotomous preferences. *Econometrica*, 72(1):257–279, 2004. 8, 20, 124

[30] Sylvain Bouveret and Michel Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems*, 30 (2):259–290, 2016. 6, 18, 20, 87

[31] Sylvain Bouveret, Katarína Cechlárová, Edith Elkind, Ayumi Igarashi, and Dominik Peters. Fair division of a graph. *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017. 28

[32] Angelina Brilliantova and Hadi Hosseini. Fair stable matching meets correlated preferences. *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems. 2022 (AAMAS 2022)*, pages 190–198, 2022. 21

[33] Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011. 2, 8, 18, 21, 31, 32

[34] Ioannis Caragiannis and Shivika Narang. Repeatedly matching items to agents fairly and efficiently. *arXiv preprint arXiv:2207.01589*, 2022. 28

[35] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–32, 2019. 2, 8, 18, 31, 74, 144

[36] Ioannis Caragiannis, Aris Filos-Ratsikas, Panagiotis Kanellopoulos, and Rohit Vaish. Stable fractional matchings. *Artificial Intelligence*, 295:103416, 2021. 8, 18, 121, 123, 128

[37] Ioannis Caragiannis, Panagiotis Kanellopoulos, and Maria Kyropoulou. On interim envy-free allocation lotteries. *Proceedings of the 22nd ACM Conference on Economics and Computation (EC 2021)*, pages 264–284, 2021. 8, 59

[38] Pak Hay Chan, Xin Huang, Zhengyang Liu, Chihao Zhang, and Shengyu Zhang. Assignment and pricing in roommate market. *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 446–452, 2016. 59

[39] Bhaskar Ray Chaudhury, Jugal Garg, and Ruta Mehta. Fair and efficient allocations under subadditive valuations. *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, 35(6):5269–5276, 2021. 144

[40] Cheng-Liang Chen, Bin-Wei Wang, and Wen-Cheng Lee. Multiobjective optimization for a multienterprise supply chain network. *Industrial & Engineering Chemistry Research*, 42(9):1879–1889, 2003. 29

[41] Xingyu Chen and Zijie Liu. The fairness of leximin in allocation of indivisible chores. *arXiv preprint arXiv:2005.04864*, 2020. 8, 18, 21, 59, 87

[42] Seyed A Esmaeili, Sharmila Duppala, Vedant Nanda, Aravind Srinivasan, and John P Dickerson. Rawlsian fairness in online bipartite matching: Two-sided, group, and individual. *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022)*, page 1583–1585, 2022. 29

[43] Alireza Farhadi, MohammadTaghi Hajiaghayi, Mohamad Latifian, Masoud Seddighin, and Hadi Yami. Almost envy-freeness, envy-rank, and nash social welfare matchings. *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, pages 5355–5362, 2021. 59

[44] Uriel Feige and Jan Vondrák. The submodular welfare problem with demand queries. *Theory of Computing*, 6(1):247–290, 2010. 8

[45] Bailey Flanigan, Paul Gölz, Anupam Gupta, Brett Hennig, and Ariel D Procaccia. Fair algorithms for selecting citizens' assemblies. *Nature*, 596(7873):548–552, 2021. 8, 21, 88

[46] Bailey Flanigan, Gregory Kehne, and Ariel D Procaccia. Fair sortition made transparent. *Advances in Neural Information Processing Systems*, 34:25720–25731, 2021. 21, 88

[47] DC Foley. Resource Allocation and the Public Sector. *Yale Economic Essays*, 7(1):73–76, 1967. 21

[48] Rupert Freeman, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. Equitable allocations of indivisible goods. *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 280–286, 2019. 28, 144

[49] Rupert Freeman, Evi Micha, and Nisarg Shah. Two-sided matching meets fair division. *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI-21)*, pages 203–209, 2021. 8, 21, 56, 85

[50] Harold N. Gabow and Robert Endre Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18(5):1013–1036, 1989. 64

[51] Yotam Gafni, Xin Huang, Ron Lavi, and Inbal Talgam-Cohen. Unified fair allocation of goods and chores via copies. *arXiv preprint arXiv:2109.08671*, 2021. 58

[52] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. 2, 6, 16, 120, 125, 136

[53] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. 63

[54] Dinesh Garg, Yadati Narahari, Earnest Foster, Devadatta Kulkarni, and Jeffrey D Tew. A groves mechanism approach to decentralized design of supply chains. *Proceedings of the 9th IEEE International Conference on E-Commerce Technology (CEC'05)*, pages 330–337, 2005. 29

[55] Jugal Garg and Setareh Taki. An improved approximation algorithm for maximin shares. *Artificial Intelligence*, 300:103547, 2021. 19, 20

[56] Joseph Geunes and Panos M Pardalos. *Supply chain optimization*, volume 98. Springer Science & Business Media, 2006. 29

[57] Mohammad Ghodsi, MohammadTaghi HajiAghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. Fair allocation of indivisible goods: Improvements and generalizations. *Proceedings of the 19th ACM Conference on Economics and Computation (EC 2018)*, pages 539–556, 2018. 19, 28, 32

[58] Alan J Goldman. Optimal center location in simple networks. *Transportation Science*, 5(2):212–221, 1971. 39

[59] Sreenivas Gollapudi, Kostas Kollias, and Benjamin Plaut. Almost envy-free repeated matching in two-sided markets. *Proceedings of the 16th International Conference on Web and Internet Economics (WINE 2020)*, pages 3–16, 2020. 8, 21, 56, 59, 85

[60] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method. *Geometric Algorithms and Combinatorial Optimization*, pages 64–101, 1988. 161

[61] Anjali Gupta, Rahul Yadav, Ashish Nair, Abhijnan Chakraborty, Sayan Ranu, and Amitabha Bagchi. Fairfoody: Bringing in fairness in food delivery. *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI 2022)*, 36:11900–11907, 2022. 29

[62] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. 49

[63] Hadi Hosseini and Andrew Searns. Guaranteeing maximin shares: Some agents left behind. *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, pages 238–244, 2021. 8, 19

[64] Hadi Hosseini, Kate Larson, and Robin Cohen. Matching with dynamic ordinal preferences. *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 936–943, 2015. 59

[65] Hadi Hosseini, Andrew Searns, and Erel Segal-Halevi. Ordinal maximin share approximation for goods. *Journal of Artificial Intelligence Research*, 74:353–391, 2022. 8, 28

[66] Hadi Hosseini, Fatima Umar, and Rohit Vaish. Two for one & one for all: Two-sided manipulation in matching markets. *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI 2022)*, pages 321–327, 2022. 8, 17

[67] Chien-Chung Huang. Cheating to get better roommates in a random stable matching. *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2007)*, pages 453–464, 2007. 17

[68] Chien-Chung Huang, Telikepalli Kavitha, Kurt Mehlhorn, and Dimitrios Michail. Fair matchings and related problems. *Algorithmica*, 74(3):1184–1203, 2016. 8, 21, 28

[69] Robert W Irving, David F Manlove, and Sandy Scott. The stable marriage problem with master preference lists. *Discrete Applied Mathematics*, 156(15):2959–2977, 2008. 87

[70] David S Johnson and Michael R Garey. *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, 1979. 27, 30, 36, 109, 111

[71] Gili Karni, Guy N Rothblum, and Gal Yona. On fairness and stability in two-sided matchings. *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, pages 92:1–92:17, 2022. 8, 21, 144

[72] Michael P Kim, Aleksandra Korolova, Guy N Rothblum, and Gal Yona. Preference-informed fairness. *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, pages 16:1–16:23, 2020. 144

[73] Bettina Klaus. "fair marriages": An impossibility. *Economics Letters*, 105(1):74–75, 2009. 21

[74] Bettina Klaus and Flip Klijn. Procedurally fair and stable matching. *Economic Theory*, 27(2):431–447, 2006. 8, 21

[75] Jon Kleinberg, Yuval Rabani, and Éva Tardos. Fairness in routing and load balancing. *Journal of Computer and System Sciences*, 63(1):2–20, 2001. 26, 29

[76] Sakunth Kumar. Over 22 lakh candidates register for JEE Main 2021, 7.32 lakh to appear for phase 4 exam. https://www.collegedekho.com/news/jee-main-2021-number-of-candidates-registered-21297/, 2021. Accessed: 2021-09-18. 86

[77] David Kurokawa, Ariel D Procaccia, and Nisarg Shah. Leximin allocations in the real world. *ACM Transactions on Economics and Computation (TEAC)*, pages 345–362, 2015. 8, 20

[78] Bo Li and Yingkai Li. Fair resource sharing and dorm assignment. *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2020)*, pages 708–716, 2020. 59

[79] Wenzheng Li and Jan Vondrák. Estimating the nash social welfare for coverage and other submodular valuations. *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 1119–1130, 2021. 28

[80] Yunpeng Li, Yichuan Jiang, Weiwei Wu, Jiuchuan Jiang, and Hui Fan. Room allocation with capacity diversity and budget constraints. *IEEE Access*, 7:42968–42986, 2019. 59

[81] Jaimie W Lien, Jie Zheng, and Xiaohan Zhong. Ex-ante fairness in the boston and serial dictatorship mechanisms under pre-exam and post-exam preference submission. *Games and Economic Behavior*, 101:98–120, 2017. 21

[82] Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On Approximately Fair Allocations of Indivisible Goods. *Proceedings of the 5th ACM Conference on Electronic Commerce (EC 2004), New York, NY, USA*, pages 125–131, 2004. 2, 8, 18, 19, 21, 31, 32, 35, 58, 74

[83] Cecilia Machado and Christiane Szerman. Centralized admission and the student-college match. Technical report, Institute of Labor Economics (IZA), 2016. 86

[84] Neeldhara Misra and Debanuj Nayak. On fair division with binary valuations respecting social networks. *Proceedings of the 8th International Conference on Algorithms and Discrete Applied Mathematics (CALDAM 2022)*, pages 265–278, 2022. 28, 29

[85] Neeldhara Misra, Chinmay Sonar, PR Vaidyanathan, and Rohit Vaish. Equitable division of a path. *arXiv preprint arXiv:2101.09794*, 2021. 28

[86] Hervé Moulin. *Fair division and collective welfare*. MIT press, 2004. 6

[87] Ashish Nair, Rahul Yadav, Anjali Gupta, Abhijnan Chakraborty, Sayan Ranu, and Amitabha Bagchi. Gigs with guarantees: Achieving fair wage for food delivery workers. *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI-22) Special Track on AI for Good*, pages 5122–5128, 2022. 29

[88] Y Narahari and Shantanu Biswas. Performance measures and performance models for supply chain decision making. *Measuring Supply Chain Performance*, 2007. 29

[89] Y Narahari and Nikesh Kumar Srivastava. A bayesian incentive compatible mechanism for decentralized supply chain formation. *Proceedings of the 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on*

Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007), pages 315–322, 2007. 29

[90] Shivika Narang and Yadati Narahari. A study of incentive compatibility and stability issues in fractional matchings. *Proceedings of 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2020)*, pages 1951–1953, 2020. 8, 165, 166

[91] Shivika Narang, Arpita Biswas, and Yadati Narahari. On achieving leximin fairness and stability in many-to-one matchings. *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022)*, pages 1705–1707, 2022. 8, 88

[92] Arvind Narayanan. Translation tutorial: 21 fairness definitions and their politics. *Proceedings of the 1st Conference on Fairness, Accountability, and Transparency (FAT* 2018)*, 1170:3, 2018. 6

[93] Thành Nguyen and Rakesh Vohra. Stable matching with proportionality constraints. *Operations Research*, 2019. 21

[94] Ciara O'Dwyer and Virpi Timonen. Doomed to extinction? the nature and future of volunteering for meals-on-wheels services. *VOLUNTAS: International Journal of Voluntary and Nonprofit Organizations*, 20(1):35–49, 2009. 26

[95] Michal Pioro. Fair routing and related optimization problems. *Proceedings of the 15th International Conference on Advanced Computing and Communications (ADCOM 2007)*, pages 229–235, 2007. 26, 29

[96] Benjamin Plaut and Tim Roughgarden. Almost envy-freeness with general valuations. *SIAM Journal on Discrete Mathematics (SIDMA)*, 34(2):1039–1068, 2020. 2, 8, 18, 19, 20, 32, 59, 87, 88

[97] Michael D. Plummer and László Lovász. *Matching Theory*. Elsevier Science, 1986. 60

[98] Ariel D Procaccia and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *Proceedings of the 15th ACM conference on Economics and computation (EC 2014)*, pages 675–692, 2014. 19

[99] H Prüfer. Neuer beweis eines satzes über permutationen. *Archiv fur Mathematik und Physik*, 27, 1918. 49

[100] Rui Qiu, Qi Liao, Renfu Tu, Yingqi Jiao, An Yang, Zhichao Guo, and Yongtu Liang. Pipeline pricing and logistics planning in the refined product supply chain based on fair profit distribution. *Computers & Industrial Engineering*, 175, 2023. 29

[101] Alvin E Roth. The economics of matching: Stability and incentives. *Mathematics of Operations Research*, 7(4):617–628, 1982. 2, 8, 17, 120, 121, 130, 132, 133

[102] Alvin E Roth. The origins, history, and design of the resident match. *JAMA: The Journal of the American Medical Association*, 289(7):909–912, 2003. 2

[103] Alvin E Roth, Uriel G Rothblum, and John H Vande Vate. Stable matchings, optimal assignments, and linear programming. *Mathematics of Operations Research*, 18(4): 803–828, 1993. 8, 17, 121, 123

[104] Alvin E Roth, Tayfun Sönmez, et al. A kidney exchange clearinghouse in new england. *American Economic Review*, 95(2):376–380, 2005. 2

[105] Uriel G Rothblum. Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54(1):57–67, 1992. 17

[106] Aitor López Sánchez, Marin Lujak, Frederic Semet, and Holger Billhardt. On balancing fairness and efficiency in routing of cooperative vehicle fleets. *Proceedings of the 12th International Workshop on Agents in Traffic and Transportation co-located with the 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI 2022)*, 3173, 2022. 29

[107] Tadeusz Sawik. On the fair optimization of cost and customer service level in a supply chain under disruption risks. *Omega*, 53:58–66, 2015. 29

[108] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., 1986. 161

[109] Jay Sethuraman, Chung-Piaw Teo, and Liwen Qian. Many-to-one stable matching: geometry and fairness. *Mathematics of Operations Research*, 31(3):581–596, 2006. 21, 121, 123

[110] Weiran Shen, Yuan Deng, and Pingzhong Tang. Coalitional permutation manipulations in the gale-shapley algorithm. *Artificial Intelligence*, 301, 2021. 17, 121

[111] Oskar Skibski. Closeness centrality via the condorcet principle. *Social Networks*, 74: 13–18, 2023. 39

[112] Walter Stromquist. How to Cut a Cake Fairly. *The American Mathematical Monthly*, 87 (8):640–644, 1980. 18, 21

[113] Zoya Svitkina and Éva Tardos. Facility location with hierarchical facility costs. *ACM Transactions on Algorithms*, 6(2):1–22, 2010. 28

[114] Chung-Piaw Teo and Jay Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operations Research*, 23(4):874–891, 1998. 8, 17, 120, 121, 123

[115] Chung-Piaw Teo, Jay Sethuraman, and Wee-Peng Tan. Gale-shapley stable marriage problem revisited: Strategic issues and applications. *Management Science*, 47(9): 1252–1267, 2001. 8, 17, 120, 121, 133

[116] Paolo Toth and Daniele Vigo. *An Overview of Vehicle Routing Problems*, chapter 1, pages 1–26. SIAM, 2002. 26

[117] Miroslaw Truszczynski and Zbigniew Lonc. Maximin share allocations on cycles. *Journal of Artificial Intelligence Research*, 69:613–655, 2020. 28

[118] Nikolaos Tziavelis, Ioannis Giannakopoulos, Rune Quist Johansen, Katerina Doka, Nectarios Koziris, and Panagiotis Karras. Fair procedures for fair stable marriage outcomes. *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, pages 7269–7276, 2020. 21

[119] Rohit Vaish and Dinesh Garg. Manipulating gale-shapley algorithm: Preserving stability and remaining inconspicuous. *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 437–443, 2017. 8, 17, 120, 121, 133

[120] Hal R Varian. Equity, Envy, and Efficiency. *Journal of Economic Theory*, 9 (1):63–91, 1974. URL https://www.sciencedirect.com/science/article/pii/0022053174900751. 21

[121] John H Vande Vate. Linear programming brings marital bliss. *Operations Research Letters*, 8(3):147–153, 1989. 17

[122] Qingyun Wu and Alvin E Roth. The lattice of envy-free matchings. *Games and Economic Behavior*, 109:201–211, 2018. 21, 22

[123] Kentaro Yahiro, Yuzhe Zhang, Nathanaël Barrot, and Makoto Yokoo. Strategyproof and fair matching mechanism for ratio constraints. *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*, pages 59–67, 2018. 21, 91, 121

[124] Yu Yokoi. Envy-free matchings with lower quotas. *Algorithmica*, 82(2):188–211, 2020. 22

[125] Dajun Yue and Fengqi You. Fair profit allocation in supply chain optimization with transfer price and revenue sharing: Minlp model and algorithm for cellulosic biofuel supply chains. *AIChE Journal*, 60(9):3211–3229, 2014. 29

[126] Yuzhe Zhang, Kentaro Yahiro, Nathanaël Barrot, and Makoto Yokoo. Strategyproof and fair matching mechanism for union of symmetric m-convex constraints. *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 590–596, 2018. 21, 91, 121

[127] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6):103–128, 2007. 72

Appendix

A Repeated Matchings

A.1 Social Welfare Maximization under Monotone Non-Increasing Valuations

We now give an alternative proof to Theorem 8.1 below. This is not a new result; we remind the reader that the proof already follows by well-known results on weighted bipartite b-matchings (see the discussion in Section 4.3.2).

Theorem 8.1 *Given a repeated instance with monotone non-increasing valuations, a repeated matching of maximum social welfare can be computed in polynomial time.*

Proof: Consider a repeated matching instance $I = \langle \mathcal{A}, \mathcal{G}, \{v_i\}_{i\in\mathcal{A}}, T\rangle$. We express the problem of computing a repeated matching of maximum social welfare as the following integer linear program:

$$
\begin{aligned}
\max \quad & \sum_{i\in\mathcal{A}} \sum_{g\in\mathcal{G}} \sum_{t=1}^{T} x_{i,g,t} \cdot v_i(g,t) & \\
\text{s.t.:} \quad & \sum_{g\in\mathcal{G}, t\in[T]} x_{i,g,t} = T, & \forall i \in \mathcal{A} \\
& \sum_{i\in\mathcal{A}, t\in[T]} x_{i,g,t} = T, & \forall g \in \mathcal{G} \\
& x_{i,g,t} \geq x_{i,g,t+1}, & \forall i \in \mathcal{A}, g \in G, t \in [T-1] \\
& x_{i,g,t} \in \{0,1\}, & \forall i \in \mathcal{A},\ g \in \mathcal{G},\ t \in [T]
\end{aligned}
$$

The binary indicator variable $x_{i,g,t}$ denotes whether agent i gets the t^{th} copy of item g ($x_{i,g,t} = 1$) or not ($x_{i,g,t} = 0$). Then, the objective is clearly to maximize the social welfare, the total value the agents get from their copies of items. The first set of constraints requires that each

agent gets exactly T copies of the items in the T rounds. The second one requires that each item is assigned in all the T rounds. The third one ensures that an agent can get her $(t+1)^{\text{th}}$ copy of an item only after she gets the t^{th} copy.

We now relax the integrality constraint by replacing $x_{i,g,t} \in \{0,1\}$ with $x_{i,g,t} \in [0,1]$. In this way, we get a linear program (LP). Well-known solvers, implementing variants of the *ellipsoid method* [60, 108], can solve this LP in polynomial time and compute an *extreme* solution. Consider such an extreme solution x and, for the sake of contradiction, assume that it is non-integral.

We first show that for every agent $i \in \mathcal{A}$ and item $g \in \mathcal{G}$, at most one variable $x_{i,g,t}$ can be non-integral. Assume otherwise for agent $i \in \mathcal{A}$ and item $g \in \mathcal{G}$, and let t_1 and t_2 be the maximum and minimum elements in set $\{t : 0 < x_{i,g,t} < 1\}$. Let $\epsilon = \min\{1 - x_{i,g,t_1}, x_{i,g,t_2}\}$ and consider the modified solution x' with $x'_{i,g,t_1} = x_{i,g,t_1} + \epsilon$ and $x'_{i,g,t_2} = x_{i,g,t_2} - \epsilon$, while x' has the same value with x on any triplet different than (i,g,t_1) and (i,g,t_2). Due to the feasibility of x, the new solution x' is clearly feasible. Furthermore, the objective value of x' is (at least) as high as that of x as it increases by $\epsilon \cdot v_i(g,t_1)$ and decreases by $\epsilon \cdot v_i(g,t_2) \leq \epsilon \cdot v_i(g,t_1)$. The last inequality follows since the valuations are monotone non-increasing. Hence, the solution x' has optimal objective value as well, and, furthermore, at least one additional integral variable compared to x: $x'_{i,g,t_1} = 1$ if $1 - x_{i,g,t_1} \leq x_{i,g,t_2}$ and $x'_{i,g,t_2} = 0$ otherwise. Thus, solution x is not extreme, a contradiction.

Now, consider the bipartite graph $G = (\mathcal{A}, \mathcal{G}, E_x)$, where E_x contains the edge (i,g) if there exists t such that $x_{i,g,t}$ has non-integer value. Observe that G contains cycles. Indeed, if G was a tree, some node u in $\mathcal{A} \cup \mathcal{G}$ would have degree 1. If $u \in \mathcal{A}$, then $\sum_{g \in \mathcal{G}, t \in [T]} x_{u,g,t}$ would include a single non-integer term (i.e., the weight $x_{u,g,t}$ of the single edge which is incident to node u). As T is integer, it would then be $\sum_{g \in \mathcal{G}, t \in [T]} x_{u,g,t} \neq T$, violating the first LP constraint for agent u. If $u \in \mathcal{G}$, then $\sum_{i \in \mathcal{A}, t \in [T]} x_{i,u,t}$ would include a single non-integer term. Again, this would imply that $\sum_{i \in \mathcal{A}, t \in [T]} x_{i,u,t} \neq T$, violating the second LP constraint for item u.

Let C be a cycle in G. Since G is bipartite, C has even length and its edges can be partitioned into two matchings M_1 and M_2. For an edge (i,g) of E_x, let $t(i,g)$ be such that $x_{i,g,t(i,g)}$ is non-integer. Also, for a set of edges M of E_x, define $V(M) = \sum_{(i,g) \in M} x_{i,g,t(i,g)} \cdot v_i(g, t(i,g))$ and, without loss of generality, assume that $V(M_1) \geq V(M_2)$. Observe that $V(M_1)$ and $V(M_2)$ are simply the contribution to the objective value by the triplets $(i,g,t(i,g))$ corresponding to the edges (i,g) of M_1 and M_2, respectively. Now let

$$\epsilon = \min\left\{1 - \max_{(i,g) \in M_1} x_{i,g,t(i,g)}, \min_{(i,g) \in M_2} x_{i,g,t(i,g)}\right\}$$

and modify the solution x to a new solution x' as follows:

- x' has the same value with x on any triplet that does not correspond to $(i, g, t(i, g))$ for an edge (i, g) of C.
- $x'_{i,g,t(i,g)} = x_{i,g,t(i,g)} + \epsilon$ for every $(i, g) \in M_1$, and
- $x'_{i,g,t(i,g)} = x_{i,g,t(i,g)} - \epsilon$ for every $(i, g) \in M_2$.

Clearly, the contribution of a triplet that does not correspond to triplet $(i, g, t(i, g))$ for an edge (i, g) of C to the objective value is the same under both solutions x and x'. The contribution from the triplets $(i, g, t(i, g))$ corresponding to edges (i, g) of M_1 increases by $\epsilon \cdot V(M_1)$ in x' compared to x, and the contribution from the triplets $(i, g, t(i, g))$ corresponding to edges (i, g) of M_2 decreases by $\epsilon \cdot V(M_2) \leq \epsilon \cdot V(M_1)$. Hence, the objective value of solution x' is also optimal. Furthermore, solution x' has at least one additional integer variable compared to x: indeed, observe that $x'_{i_1,g_1,t(i,g)} = 1$ for some edge (i_1, g_1) of M_1 if $1 - \max_{(i,g)\in M_1} x_{i,g,t(i,g)} \leq \min_{(i,g)\in M_2} x_{i,g,t(i,g)}$ and $x'_{i_2,g_2,t(i_2,g_2)} = 0$ for some edge (i_2, g_2) of M_2, otherwise. Thus, solution x is not extreme, again a contradiction. □

A.2 Maximizing Social Welfare over Repeated Allocations

Observe that when we do not have the constraint that each agent must get exactly one good in each round, that is, the more general allocation setting, we can make decisions for each good independently. Further, from the definition of the value for an allocation, for the purposes of social welfare, we are indifferent between all allocations who allocate each good to each agent the same number of times. Hence, given an allocation A, define allocation A' which allocates each good $g \in \mathcal{G}$ to agent 1 for rounds $1, \cdots, N(A_1, g)$, to agent 2 for rounds $N(A_1, g)+1, \cdots, N(A_1, g)+N(A_2, g)$ and similarly to agent i for rounds $\left(\sum_{i'=1}^{i-1} N(A_{i'}, g)\right) + 1, \cdots, \sum_{i'=1}^{i} N(A_{i'}, g)$. Observe that for each $i \in \mathcal{A}$, $v_i(A') = v_i(A)$ and hence the two allocations also have the same social welfare. Henceforth, we shall only construct allocations which allocate each good to agents in order. Observe that for such allocations, the last round in which the good is matched to agent $n - 1$ determines agent n's allocation.

We shall now show how to find a social welfare maximizing allocation of an good. For each good, we shall define a directed acyclic graph. There will be a special source node α and a sink node β, all the remaining nodes will belong to one of $n-1$ "layers". Edges will only go from the i^{th} layer to the $(i+1)^{\text{th}}$ layer, from the source to the first layer or from the last layer to the sink. There are no edges within any layer. As a result, any path in this graph must

contain at most one node from each layer. In particular, any path from α to β must contain exactly one node from each layer. Every path from α to β shall correspond to an allocation. Layer i corresponds to agent i, and the T nodes each correspond to the last round in which i will be allocated the good. We shall set up the edge weights so that a longest $\alpha - \beta$ path shall correspond to a social welfare maximizing allocation of the good.

Given good g, we define the directed acyclic graph $G = (V, E)$ as follows: $V = \{\alpha, \beta\} \cup \{x_{i,t} : i \in [n-1] \text{ and } t \in [T]\}$ and $E = \{(\alpha, x_{1,t}) : t \in [T]\} \cup \{(x_{i,t}, x_{i+1,t'}) : i \in [n-1], t, t' \in [T] s.t. t \leq t'\} \cup \{(x_{n-1,t}, \beta) : t \in [T]\}$. We define weights on the edges as follows: for $e = (\alpha, x_{1,t})$, set $w_e = \sum_{t'=1}^{t} v_1(g, t')$, for $e = (x_{i,t_1}, x_{i+1,t_2})$ with $i \in [n-2]$, $t_1, t_2 \in [T]$ with $t_1 \leq t_2$, set $w_e = \sum_{t'=1}^{t_2 - t_1} v_i(g, t')$ and for $e = (x_{n-1,t}, \beta)$, set $w_e = \sum_{t'=1}^{T-t} v_n(g, t')$.

Observe that there are no edges from $x_{i,t}$ to $x_{i+1,t'}$ where $t' < t$. Hence, given an $\alpha - \beta$ path P, let t_i^P be such that x_{i,t_i^P} is the node in layer i which is in P. For any $i \in [n-2]$, $t_i^P \leq t_{i+1}^P$. Define an allocation of good g, $A(P, g)$ where agent i is allocated g $t_i^P - t_{i-1}^P$ times for $i \in [n-1]$ and agent n is allocated g $T - t_{n-1}^P$ times. Here, the value of agent $i \in [n-2]$ for this allocation is $v_i(A_i(P, g)) = \sum_{t=1}^{N(A_i(P,g),g)} v_i(g, t) = \sum_{t=1}^{t_i^P - t_{i-1}^P} v_i(g, t) = w_{x_{i-1,t_{i-1}^P}, x_{i,t_i^P}}$. For agent n, $N(A_n(P, g), g) = T - t_{n-1}^P$ and $v_n(A_n(P, g)) = \sum_{t=1}^{T - t_{n-1}^P} v_n(g, t) = w_{x_{n-1,t_{n-1}^P}, \beta}$. Thus, $SW(A(P, g)) = \sum_{e \in P} w_e$. Thus, for every $\alpha - \beta$ path of length l, we have an allocation of social welfare l.

Now, for any allocation of goods, we can create an allocation with equal social welfare by allocating each good to agents in order. Further, for the allocation setting, each good may be allocated independently. As a result, for every allocation of a single good, we can create a path in the corresponding directed graph with length equal to the social welfare. Hence, we can find a social welfare maximizing allocation by finding a longest $\alpha - \beta$ path for each good g.

B Fairness and Stability in Many-to-one Matchings

B.1 Other Fairness Notions

Envy and Stability: Fairness has been widely studied in computational social choice, with various fairness notions considered for both divisible and indivisible goods. Since we are interested in integral matchings (matchings where agents are matched wholly/ integrally, no agent is matched partially/fractionally), we shall only discuss allocations of indivisible goods. Often, the first fairness notion that comes to mind when we think of fair allocations is *envy-freeness* (EF). Informally, an EF allocation guarantees that every agent would prefer its own allocation over any other agent's allocation. Unfortunately, EF allocations/matchings need not exist in indivisible settings. Consider $\mathcal{S} = \{s_1, s_2\}$ and $\mathcal{C} = \{c_1, c_2\}$ with isometric valuations $V_{11} = 10$, $V_{12} = 5$, $V_{21} = 8$, and $V_{22} = 2$. Let μ be a matching such that $\mu(s_1) = c_1$ and $\mu(s_2) = c_2$. Since $u_2(c_1) > u_2(c_2)$, the agent s_2 envies s_1, similarly, agent c_2 envies c_1. It is easy to see that in every possible matching μ, there will always be at least one agent who will envy the other agent.

Envy-freeness up to one item (EF1) is a popular notion of fairness for allocations of indivisible items. An allocation $A = (A_1, \cdots, A_n)$ is said to be EF1 if for every distinct pair of agents i and j, there exists $g \in A_j$ such that $u_i(A_i) \geq u_i(A_j \backslash \{g\})$. An allocation is *envy-free up to any good* (EFX) if for all $g \in A_j$, $u_i(A_i) \geq u_i(A_j \backslash \{g\})$. In one-to-one matchings, EF1 and EFX are achieved trivially.

In the case of many-to-one matchings, we can find matchings that are EF1 for the colleges by using a round robin procedure. However, such a matching need not be stable, even under ranked isometric valuations. In fact, there exist ranked isometric valuations instances where no matching simultaneously satisfies stability and EF1. Consider the following example: let $n = 4$ and $m = 2$. The valuation matrix is as in Table 8.1. Now it is easy to verify that

student	c_1	c_2
s_1	100	10
s_2	99	9
s_3	20	4
s_4	19	3

Table 8.1: Valuation Matrix

c_1	c_2	$E_{\mathcal{S}}$	$E_{\mathcal{C}}$	E_{total}
1-4	-	0	26	26
1-3	4	16	20	36
1,2	3,4	32	12	44
1	2-4	120	38	158
-	1-4	0	238	238

Table 8.2: Stable Matching Space

A counterexample for Envy and Stability

the only EF1 matching is $\mu = \{(s_1, c_1), (s_2, c_2), (s_3, c_1), (s_4, c_2)\}$. This however is not stable

as (s_2, c_1) form a blocking pair. Hence, when stability is non-negotiable, EF1 cannot be the fairness notion of choice for isometric valuations. Consequently, neither can EFX.

It has been shown that the space of stable matchings of any given instance can be captured as the extreme points of a linear polytope. Thus, we can optimize any linear function over this space. Consequently, our next idea may be to look for a stable matching that minimizes average envy, or equivalently, total envy. But that too can often lead to matchings that are inherently unfair. Consider the example given in Table 8.1. The stable matchings in this example and the envy they induce is as in Table 8.2.

Clearly, in this example, matching all the students to c_1 reduces the total/average envy but this is obviously unfair to c_2. This is happening despite the fact that there are more students than there are colleges. Note that in this example, there is in fact a ranking and yet envy doesn't work well. In fact, this example can be extended for much larger values of n so that c_2 is matched to no students in the stable matching which minimizes total envy.

Welfare Functions based Fairness: Maximizing for Nash Social Welfare over the space of stable matchings for such examples would again result in matchings where c_2 is matched to exactly 1 student even for very high values of n. These solutions in settings like labour markets or college admissions would result in the rich getting richer, defeating the purpose of fairness. One alternative would be egalitarian welfare, where we aim to simply maximize the valuation of the worst off agent. Clearly, all leximin optimal solutions would also optimize for egalitarian welfare. However, in the case of ranked isometric valuations, as a result of Lemma 5.1, any fair and stable matching must match s_n to c_m, and s_n will always have least valuation. As a result, all complete stable matchings under isometric valuations will optimize egalitarian welfare. Thus, in this setting, egalitarian welfare alone is not enough to ensure true fairness. In a similar spirit, a good approximation to egalitarian welfare may be satisfied by matchings which give no guarantee to the remaining agents, and can even be inefficient to a large extent. In order to avoid such outcomes, we study leximin optimal fairness.

B.2 Incentive Compatibility

We now explore the existence of incentive compatible mechanisms which return the leximin optimal over the space of stable matchings. Given our results on the tractability of this problem, we restrict our focus to instances with ranked valuations. Observe that an impossibility here also implies an impossibility when the valuations are unrestricted.

Narang and Narahari[90] study the existence of incentive compatible mechanisms which find stable fractional matchings when $m = n$. In particular, they study a class of matchings instances where incentive compatibility is achievable. Interestingly the space of instances

with rankings is subsumed by this class for one-to-one matchings. We show that for any mechanism that computes the leximin optimal stable matching, agents have an incentive to be honest if and only if $n = m$, in which case, we are essentially in the setting studied by Narang and Narahari[90].

Lemma 8.1 *A mechanism that takes as input an SMO instance I with general ranked valuations and outputs the leximin optimal stable matching does not give any agent an incentive to misreport their valuations if and only if $n = m$.*

Proof: We first begin with the simpler case. Under strict ranked valuations, if $n = m$ then there is a unique stable matching, irrespective of the exact valuations reported. Thus, no agent has an incentive to misreport their preferences.

Now we look at the case when $n > m$ with rankings. We shall show that there always exists an agent who wishes to misreport their valuations.

Given $I = \langle \mathcal{S}, \mathcal{C}, U, V \rangle$, let $Best : \mathcal{S} \to \mathcal{C}$ be a function such that $Best(s_i)$ is the highest ranked college , s_i can be matched to in a stable matching where each agent's matching is non-empty. Similarly let $Worst$ be a a function such that $Worst(c_j)$ is the lowest ranked student , c_j can be matched to in a stable matching where each agent's matching is non-empty.

Therefore $Best(c_j) = s_j$ for all $j \in [m]$. For $i \leq n-m+1$, $Best(s_i)c_1$ and for $i \geq n-m+1$, $Best(s_i) = c_{m-n+i}$. Thus if $i \neq 1$, $Best(s_i) \neq Best^{-1}(s_i)$.

Let μ^* be the leximin optimal stable matching for I. Fix p such that $1 < p \leq m < n$

Case 1: $\mu^*(s_p) = Best(s_p)$.

Let $c_t = Best^{-1}(s_p)$. Define $\gamma = \min\{u_i(c_j) |\, u_i(c_j) > 0\} \cup \{v_j(s_i) |\, v_j(s_i) > 0\}$

Set $\alpha = \left\lceil \frac{v_t(s_p)}{\gamma} \right\rceil n$. Define v' where

$$v'(s_i) = \begin{cases} v_t(s_i) & i > p \\ v_t(s_i)/\alpha & i \leq p \end{cases}$$

Now c_t can misreport their valuation function as v' and ensure they are matched to $\{s_p, \cdots, Worst(c_j)\}$. This is because our choice of v' ensures that c_t will be considered as the agent with the lowest valuation. Thus c_t has an incentive to misreport

Case 2: $\mu^*(s_p) \neq Best(s_p)$.

Let $\delta = \min_{j>1} v_j(Worst(c_j))$. Recall that if $u_p(Best(s_p)) < \delta$ then in the leximin optimal stable matching, s_p must be matched to $Best(s_p)$. Thus, s_p can misreport u_p as u' where

$u'(c_j) = \frac{\delta - \epsilon}{j}$, $\epsilon > 0$.

This ensures that s_p is matched to $Best(s_p)$ by misreporting. Thus, s_p has an incentive to misreport.

□

B.3 Capacity Constrained Settings

In FaSt and FaSt-Gen we had assumed that each college has capacity $b_j = n - 1$. We now give the formal algorithms for when this assumption is relaxed to allow colleges to have arbitrary capacity. The Demote procedure (Algorithm 5.1) remains the same as the uncapacitated setting.

B.3.1 Ranked Isometric Valuations

We first translate FaSt for capacity constrained settings. Recall that $b_j \in [n]$ denotes the capacity of c_j. We assume without loss of generality that $b_j \geq 1$. Here, the student optimal stable matching, matches c_1 to as many students as possible, i.e. $\min(n - m + 1, b_1)$. If $b_1 < m - n + 1$, then we match as many students to c_2 as possible i.e. $\min(n - b_1 - m + 2, b_2)$. This process is now repeated till all the students are matched. Now we can simply follow FaSt as is, taking students from the lowest ranked college with multiple students matched to it, with the additional constraint that we fix a college when it reaches full capacity.

In the tie-breaking routine, we must also check if the capacity constraint is violated in the while loop. Now, μ will still continue to be a valid matching as we only update μ is we find a leximin increase. This update happens only after checking that μ' is still a valid matching in the while loop. As a result, μ is always a valid matching.

Despite the additional capacity constraints, the five observations listed in Section 5.3.1 continue to hold. Thus by analogous reasoning to the uncapacitated setting, we can optimize the leximin tuple, one entry at a time. Thus, CapFaSt correctly finds the leximin optimal stable matching, from a similar argument as Theorem 5.1.

Algorithm 8.1: CapFaSt

Input: Instance of ranked isometric valuations with capacities $\langle \mathcal{S}, \mathcal{C}, V, B \rangle$
Output: μ

Initiate a stable matching: μ as the student optimal stable matching;
Initialize $i \leftarrow n-1$, $down \leftarrow m$, $up \leftarrow \max\{j \in [m] || \mu(c_j)| > 1\}$;
Set $\mathcal{L}$ as the leximin tuple for μ;
Set $pos[i]$ as the position of $\mathcal{A}_i$ in $\mathcal{L}$, $i \in [n]$;
Initialize $\mathcal{F} \leftarrow \{\mathcal{A}_n\}$; // stores the agents whose matching is fixed.
while $i > down - 1$ *AND* $down > 1$ **do**
  if $v_{down}(\mu) \geq v_{i(down-1)}$ *OR* $|\mu(j)| = b_j$ **then**
    $down \leftarrow down - 1$;
  else
    if $[v_{i\,down} > v_{down}(\mu)]$ **then**
      $\mu \leftarrow Demote(\mu, i, up, down)$;
    else
      if $v_{i\,down} < v_{down}(\mu)$ **then**
        $down \leftarrow down - 1$;
      else
        $k \leftarrow i - 1$;
        $t \leftarrow pos[i]$;
        $\mu' \leftarrow Demote(\mu, i, up, down)$;
        while $k > down - 1$ *AND* $|\mu'(down)| \leq b_j$ **do**
          if $u_{k\,down} > \mathcal{L}[t]$ **then**
            $\mu \leftarrow Demote(\mu', k, up, down)$;
            $i \leftarrow k$;
            break;
          else
            if $v_{ij} < v_j(\mu)$ **then**
              $down \leftarrow jdown - 1$;
              break;
            else
              $\mu' \leftarrow Demote(\mu', k, up, down)$;
              $k \leftarrow k - 1$, and $t \leftarrow t + 1$;
        if $k = down - 1$ *AND* $\mu \neq \mu'$ **then**
          $down \leftarrow down - 1$;
  $\mathcal{F} \leftarrow \{\mathcal{A}_i, \cdots, \mathcal{A}_n\} \cup \{c_{j+1}, \cdots c_m\}$;
  $Update(\mathcal{L}, \mu, pos)$;
  $i \leftarrow i - 1$;
  if $(|\mu(c_{up})| = 1$ *OR* $down = up)$ *AND* $(up > 1)$ **then**
    $up \leftarrow \max\{j < up\, n |\, \mu(c_j)| > 1\}$

B.3.2 General Ranked Valuations

We now translate FaSt-Gen for capacity constrained settings. We again start with the student optimal complete stable matching. This is defined in the preprocessing routine given in Algorithm 8.2. This matches as many students as possible to c_1 then if there are more than $m-1$ students unmatched, as many as possible to c_2 and so on. Note that the fixing carried out after defining the matching is not necessary for the correctness of the algorithm, and the algorithm would still continue to correctly compute the leximin optimal stable matching if we did not perform this. In the main algorithm in Algorithm 8.3, we essentially run the main while loop in FaSt-Gen multiple times. We must do this because a college that is at full capacity may have given out some students after having the upper limit fixed. In such a case we may see a leximin increase by adding more students to this college .

As a result, in Algorithm 8.3, we unfix the upper limit of all such colleges who after having been at full capacity, gave out some students , till such time that there are no longer any such colleges . Since we never add more students to a college at full capacity (due to the if condition in line 12 of Algorithm 8.3), μ continues to be a valid matching. Now for the look ahead routine detailed in Algorithm 8.4, the only college to which more students are added is c_{down} and the condition of the while loop ensures that it is never overfilled. The correctness of the capacitated version of the algorithm follows from the correctness of FaSt-Gen.

One point that needs scrutiny is the running time. While it may appear that the running time may become uncontrolled, note that a particular student and college pair are only ever considered in one particular execution of the first while loop, in which case they can be consider up to $m-1$ times. The comparison of a leximin tuple also continues to take $O(n)$ time. As a result, the running time continues to be $O(m^2n^2)$.

Algorithm 8.2: Preprocessing

Input: Instance of general ranked valuations $\langle \mathcal{S}, \mathcal{C}, U, V, B \rangle$
Output: μ, $UpperFix$, $LowerFix$, $Unfixed$

//Initiate a stable matching with the student optimal complete stable matching: ;
$j \leftarrow 1$;
$p \leftarrow n - m + 1$;
$t \leftarrow 0$;
while $j \leq m$ **do**
    $i \leftarrow \min\{b_j, p\}$;
    $\mu(c_j) \leftarrow \{\mathcal{A}_{t+1}, \cdots, \mathcal{A}_{t+i}\}$;
    $t \leftarrow t + i$;
    if $i < p$ **then**
        $p \leftarrow (p - i) + 1$;
    else
        $p = 1$;
    $j \leftarrow j + 1$;
$UpperFix \leftarrow \{c_1\},\ LowerFix \leftarrow \{c_m\}$;
Set $\mathcal{A}_i$ to be the highest ranked student s.t. $u_i(\mu) \leq v_j(\mu)$ for all $j \in [m]$, $|\mu(\mu(\mathcal{A}_i))| = 1$ and $i \neq 1$;
if $i < n$ **then**
    $c_j \leftarrow \mu(\mathcal{A}_i)$;
    $UpperFix \leftarrow UpperFix \cup \{c_j, \cdots, c_m\}$;
    $LowerFix \leftarrow LowerFix \cup \{c_j, \cdots, c_m\}$;
$SoftFix \leftarrow \emptyset$;
$Unfixed \leftarrow UpperFix^c$;

Algorithm 8.3: CapFaSt-Gen

Input: Instance of general ranked valuations with $\langle \mathcal{S}, \mathcal{C}, U, V, B \rangle$
Output: μ

$\langle \mu, UpperFix, LowerFix, Unfixed \rangle \leftarrow Preprocessing(\langle \mathcal{S}, \mathcal{C}, U, V, B \rangle)$;
$Flag \leftarrow True$;
while *Flag* **do**
  Set $T[j] \leftarrow 0$ for all $j \in [m]$;
  while $|UpperFix \backslash LowerFix| + |LowerFix| < m$ **do**
    $up \leftarrow \min_{j \notin LowerFix} j$ and $down \leftarrow \text{argmin}_{j \in Unfixed} v_j(\mu)$.;
    $SoftFix \leftarrow SoftFix \backslash \{(j, j') | j' \leq up < j\}$;
    if $|\mu(c_{up})| = 1$ *OR* $v_{up}(\mu) \leq v_{down}(\mu)$ **then**
      $LowerFix \leftarrow LowerFix \cup \{c_{up}\}$;
    else
      if $|\mu(c_{down})| = b_{down}$ **then**
        $UpperFix \leftarrow UpperFix \cup \{c_{down}\}$
      else
        $\mu' \leftarrow Demote(\mu, down, up)$;
        if $\mathcal{L}_{\mu'} \geq \mathcal{L}_{\mu}$ **then**
          if $|\mu(c_{up})| = b_{up}$ **then** $T[up] = 1$;
          $\mu \leftarrow \mu'$;
        else
          if $sourceDec(\mu', \mu) = c_{up}$ **then**
            $LowerFix \leftarrow LowerFix \cup \{c_{up}\}$ and
             $UpperFix \leftarrow UpperFix \cup \{c_{up+1}\}$;
          else
            if $sourceDec(\mu', \mu) \in \mathcal{S}$ **then**
              $c_t \leftarrow \mu(sourceDec(\mu', \mu))$;
              $LowerFix \leftarrow LowerFix \cup \{c_t\}$;
              $UpperFix \leftarrow UpperFix \cup \{c_{t+1}\}$;
              $A \leftarrow \{j | j > t+1, j \in Unfixed\}$;
              $SoftFix \leftarrow SoftFix \cup (A \times \{t+1\})$;
            else
              $(\mu, LowerFix, UpperFix, SoftFix) \leftarrow$
               $LookaheadRoutine(\mu, down, LowerFix, UpperFix, SoftFix)$;
    $Unfixed \leftarrow \{j | j \notin UpperFix$ or $(j, j') \notin SoftFix$ for some $j' > j\}$;
if $t[j] = 0$ *for all* $j \in [m]$ **then**
  $Flag \leftarrow False$;
else
  Set j to be the highest ranked college such that $T[j] = 1$;
  $UpperFix \leftarrow \{c_1, \cdots c_{j-1}\}$ and $LowerFix \leftarrow \{c_m\}$;

Algorithm 8.4: Look ahead Routine

Input: $I, \mu, down, LowerFix, UpperFix, SoftFix$
Output: $\mu, LowerFix, UpperFix, SoftFix$

$\langle \mu', LF, UF \rangle \leftarrow \langle \mu,\ LowerFix,\ UpperFix \rangle$;
while *($|LF| + |UF \backslash LF| < m$) AND ($|\mu(c_{down}) < b_{down}|$)* **do**
  $up \leftarrow \min_{j \notin LowerFix} j$;
  if $|\mu(c_{up})| = 1$ *OR* $v_{up}(\mu) \leq v_{down}(\mu)$ **then**
    $LF \leftarrow LF \cup \{c_{up}\}$;
  else
    $\mu' \leftarrow Demote(\mu', up, down)$;
    if $\mathcal{L}(\mu') \geq \mathcal{L}(\mu)$ **then**
      $\mu \leftarrow \mu'$;
      $LoweFix \leftarrow LF$, $UpperFix \leftarrow UF$;
      break;
    else
      //Decrease in leximin value, need to check the source of the decrease;
      if $sourceDec(\mu', \mu) = c_{up}$ **then**
        $LF \leftarrow LF \cup \{c_{up}\}$, $UF \leftarrow UF \cup \{c_{up+1}\}$;
      else
        if $sourceDec(\mu', \mu) \in \mathcal{S}$ **then**
          $c_t \leftarrow \mu'(sourceDec(\mu', \mu))$;
          if $t = down$ **then**
            //Cannot increase leximin value due to c_{down};
            $UpperFix \leftarrow UpperFix \cup c_{down}$;
          else
            $SoftFix \leftarrow SoftFix \cup (down, t)$;
          break;