ABSTRACT

Title of Dissertation: THE PRICE OF FAIRNESS IN ALGORITHMIC DECISION-MAKING

Max Stephen Springer
Doctor of Philosophy, 2025

Dissertation Directed by: Professor MohammadTaghi Hajiaghayi
Department of Computer Science

Algorithms play a fundamental role in modern decision-making, impacting economic structures, cultural landscapes, and resource allocation at unprecedented scales. While optimization techniques often prioritize efficiency—finding solutions quickly with minimal resources—many real-world applications require an additional consideration: fairness. This thesis explores the intricate trade-offs between efficiency and fairness in algorithm design, focusing on two key domains: fair division and fair hierarchical clustering.

In the first part, I examine the fair division problem, which seeks to distribute resources among stakeholders in an equitable and efficient manner. I introduce novel approaches for balancing fairness constraints with computational feasibility, investigating both offline and online settings. In offline fair division, I analyze classical notions such as envy-free allocations and extend them to more general settings, including weighted agent priorities and multigraph valuation models. In the online setting, where decisions must be made without full knowledge of future

arrivals, I present new algorithms for fair resource allocation, including solutions for the online Santa Claus problem and class-fair bipartite matching.

The second part of this thesis focuses on fair hierarchical clustering, a problem central to machine learning and data analysis. Many clustering algorithms reinforce biases in data representation, disproportionately impacting marginalized groups. I develop novel fair clustering algorithms that balance demographic representation while preserving computational efficiency. My work introduces the first explainable algorithm for fair hierarchical clustering and provides near-optimal polylogarithmic approximation guarantees, significantly improving upon prior results.

By leveraging techniques from combinatorial optimization, game theory, and online algorithms, this thesis provides theoretical insights and practical methodologies for designing algorithms that uphold fairness constraints without sacrificing efficiency. These contributions have broad implications for resource allocation, machine learning, and algorithmic fairness in large-scale systems.

The Price of Fairness in Algorithmic Decision Making

by

Max Stephen Springer

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:
Professor MohammadTaghi Hajiaghayi, Chair & Advisor
Professor Subramanian Raghavan, Dean's Representative
Professor William Gasarch
Professor Deep Ray
Professor John P. Dickerson

Dedication

This thesis is dedicated to my incredible parents and siblings. Without their endless support, this whole journey would not have been possible. That being said, I expect none of them to read beyond this point.

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost, I'd like to thank my advisor, Professor MohammadTaghi Hajiaghayi, for giving me all the tools to succeed as a computer scientist, professor, and mentor. I owe my professional career and deep passion for discrete mathematics to him.

I'd like to thank all my co-authors from across the globe: Hannaneh Akrami, Kiarash Banihashem, MohammadHossein Bateni, Neslihan Bulut, Diptarka Chakraborty, Vincent Cohen-Addad, John Dickerson, Iman Gholami, Shayan Chashm Jahan, MohammadReza Khani, Marina Knittel, Dariusz Kowalski, Mohammad Mahdavi, Ruta Mehta, Jan Olkowski, Debmalya Panigrahi, Suho Shin, and Hadi Yami. The bulk of this research would not have been possible without our discussions.

Beyond College Park, I also had the great fortune to work with a number of brilliant scientists throughout industry and government labs. A very special thank you to Sasan Tavakkol, Matthew Andrews, Atefeh Mohajeri, Arthur Sherman, Hal Blumenfeld, Aya Khalaf, Heinz Krestel, and Orit Lavi.

My academic family from my past life as a computational neuroscientist, without whom I probably would have been too chicken to apply for a PhD: Ganesh, Julia, Noah, Peter, Reese, and the late great Peter Salvino, thank you for the encouragement and support.

I naturally also need to thank my friends who have been perplexed by my decision to stay a student (essentially) indefinitely: Bryce, Cameron, Chris, Claire, Giacomo, Jinghan, Michael, Neel, Norman, Shashank, Tom, and Zac.

Last, but certainly not least, thank you to my family: Mother, Father, Brother, Sister, Mitch, Asa, Caroline, Harper, Duckworth, Goose Wayne, and my loving partner Reshmasai.

–

I would lastly like to acknowledge financial support from the National Science Foundation's Graduate Research Fellowship (Grant No. DGE 1840340) for all the projects discussed herein.

All the people we used to know,

they're an illusion to me now.

Some are mathematicians,

some are carpenters' wives.

Don't know how it all got started.

I don't know what they're doin' with their lives.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The notion of an "algorithm" has evolved from being merely a step-by-step procedure to a powerful, pervasive entity that shapes economies [Burrell and Fourcade, 2021, Latzer et al., 2016, Slaughter et al., 2020] and, perhaps most remarkably, influences our cultural landscapes [Chayka, 2025, Gillespie, 2016, Seaver, 2017]. These algorithms derive much of their power from the ever-growing amounts of data generated and transferred each day, currently estimated to be about 402.74 *million terrabytes* [Taylor, 2024]. However, while the abundance of data has made optimization problems easier in certain aspects, it also raises critical challenges. Namely, how can we efficiently use this data at scale while ensuring that optimization processes remain fair to all stakeholders?

This thesis address one such central question in modern algorithm design: to what extent can optimization metrics be balanced with fairness constraints without sacrificing efficiency? In particular, my research investigates the inherent trade-offs and opportunities between efficiency—the ability to compute optimal or near-optimal solutions quickly and with minimal resources—and fairness, which ensures that outcomes do not unduly disadvantage any group or individual.

Throughout this work, I examine scenarios where these two goals are compatible, and others where a competitive tension arises between them. Consider a simple yet illustrative example:

splitting the bill at a group dinner. The efficient solution is to divide the total cost equally among all diners. However, this approach benefits those who consumed more at the expense of those who consumed less. Conversely, an equitable solution involves allocating costs based on individual consumption—a tedious process that requires more effort but is guaranteed to yield a fairer outcome. Though seemingly trivial, problems of this nature lie at the heart of algorithmic decision-making in complex systems, where competing objectives often arise between fairness and efficiency.

This thesis offers a broad exploration of algorithmic problems where these dual objectives are central. I analyze the ways in which efficient solutions can be achieved while mitigating potential imbalances in outcomes. The work focuses on two major areas of study: resource division and data clustering.

At first glance, these two problem domains may seem orthogonal. However, I demonstrate that both can be approached using a common combinatorial framework, allowing for general principles to emerge across applications. This combinatorial toolbox leverages concepts from optimization, game theory, and graph theory to address real-world challenges faced by algorithm designers today.

In the following sections, I introduce each of these key research areas in detail. I motivate their relevance in modern computing systems, explain the interplay between fairness and efficiency in these contexts, and summarize my contributions toward addressing these challenges. Ultimately, this thesis aims to provide both theoretical insights and practical methodologies for designing algorithms that respect fairness constraints while remaining efficient at scale.

1.1 Fair Division

The problem of distributing resources equitably among stakeholders is a cornerstone of algorithmic game theory and economics. Although the mathematical study of fair division began with Hugo Steinhaus's seminal work in 1948 [Steinhaus, 1948], the concept has long been essential for resolving disputes such as inheritance settlements or the dissolution of partnerships.

With the advent of the Internet in the early 1990s, peer-to-peer interactions began to proliferate, creating new demands for fair division solutions. These interactions span the allocation of computing resources, data, reputation scores, and other intangible assets [Moulin, 2019]. As modern systems increasingly depend on equitable resource sharing, insights from the fair division literature have become vital to ensuring that these interactions are both efficient and just.

In essence, the fair division problem seeks to allocate a set of (indivisible or divisible) goods among a group of agents (stakeholders) in a way that satisfies both fairness and efficiency (see Figure 1.1). A classic example is the cake-cutting problem [Brams and Taylor, 1996, Brams and Taylor, 1995, Robertson and Webb, 1998], which asks how two people can divide a cake such that neither envies the other's portion. This problem has a deceptively simple solution: one person cuts the cake into two pieces they believe to be equal, and the other person selects the piece they prefer. This "I-cut-you-choose" algorithm is both intuitive and remarkably robust, serving as the foundation for numerous fair division algorithms developed to handle far more complex scenarios.

These algorithms are essential in modern settings where resources are limited but demand is high, such as allocating public goods [Moulin, 2004], managing bandwidth in networks [Kelly et al., 1998], or distributing medical supplies during shortages [Persad et al., 2009]. Users can

even consult these algorithms directly for everyday tasks: Spliddit [Goldman and Procaccia, 2015] is a popular online platform that implements fair division algorithms to help equitably divide rent, goods, and credit. Similarly, the Kajibuntan app [Igarashi and Yokoyama, 2023], launched in 2023, allows housemates to allocate household chores according to their preferences using fairness-optimized algorithms.

Despite these successes, the fair division problem is far from fully resolved. Many positive results have been obtained under specific fairness definitions or restrictive assumptions, yet numerous fundamental questions remain open. In this thesis, I explore both classical and modern challenges in fair division, focusing on two key settings: offline fair division, where all resources and agents are known in advance, and online fair division, where resources and requests arrive sequentially without full knowledge of the future.

1.1.1 Offline Computation Model

In the offline setting, all resources and stakeholders are known in advance. The classic cake-cutting example discussed earlier is a representative model of this framekwork. Within this setting, a variety of solution concepts for fairness exist, though my thesis focuses primarily on the pairwise *envy-free* (EF) condition—where no agent prefers another agent's allocation to their own.

The general form of the envy-free condition is known to be NP-hard, as demonstrated by a straightforward reduction from the PARTITION problem [Lipton et al., 2004], and trivial examples showcase an unbounded approximate guarantee. Due to this computational intractability, much of the research has shifted toward relaxed fairness concepts, particularly envy-free up to any

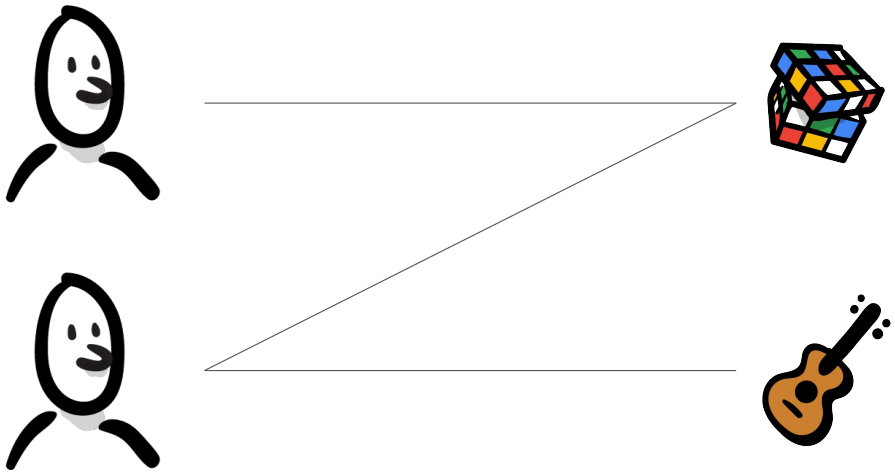


Figure 1.1: A fair allocation instance with two agents and two items. In this offline setting, where all items and their potential allocation are known, the optimal solution of giving the first item to the first agent and second item to the second agent is trivially computed.

good (EFX) allocations. Under EFX, an agent may still envy another's allocation, but the envy must disappear if a single item is removed from the envied allocation [Caragiannis et al., 2019b, Farhadi et al., 2021]. Despite its broad appeal, EFX remains an open problem: the best-known approximation guarantee is 0.618, and the optimal solution has been described as "fair division's most enigmatic question" [Procaccia, 2020]. In this thesis, I extend the study of EFX into new, generalized domains, to push the boundaries of its theoretical existence and practical applications.

Asymmetric Agent Priorities. One line of research extends EFX to weighted fair division, where stakeholders have asymmetric priorities or entitlements to resources [Springer et al., 2024]. For example, when distributing health supplies among countries, it may be necessary to weight allocations based on population size to avoid giving equal resources to both sparsely and densely populated nations. Such considerations were crucial in the distribution of COVID-19 vaccines, highlighting a highly relevant and complex fair division problem [Fallucchi et al., 2021,Laventhal

et al., 2020, Munguía-López and Ponce-Ortega, 2021].

The weighted fair division problem generalizes the standard problem by introducing variable priorities, thus inheriting and amplifying its complexity. This prompts a central question: how much more difficult does fair division become when weights are introduced?

Our findings reveal that in cases where weights have a common divisor, the problem can be reduced to a series of replicated instances of the unweighted problem. Standard algorithms can then be extended to handle these instances without significant changes. However, when weights are coprime or drawn from real-valued domains, the problem diverges significantly. In fact, even for just two parties, weighted EFX (WEFX) is not always guaranteed to exist. Furthermore, the algorithmic approximation guarantee worsens, scaling with the input size rather than remaining a constant factor.

Multigraph Valuations. In a separate line of work, I explore fair division under graphical valuation models, introduced by [Christodoulou et al., 2023]. In this setting, agents are represented as nodes in a graph, and edges indicate a shared interest in particular resources. This model captures resource competition between agents and is applicable to scenarios where allocations must respect inter-agent dependencies (e.g. sharing limited bandwidth across connected users).

While simple graph structures admit EFX solutions under certain assumptions, they fail to capture the full complexity of real-world fair division instances, where agents often compete over multiple overlapping resources. To address this limitation, I generalize the model to multigraphs, where multiple edges between nodes represent multiple shared interests.

This generalized setting presents combinatorial challenges similar to those seen in weighted fair division.

When two agents share multiple resources, their interaction can be viewed as a smaller instance of the standard fair division problem. By solving these smaller problems using known algorithms and applying local updates to the multigraph, we develop a method to iteratively resolve unfair allocations across the graph.

Our results show that while EFX allocations do not inherently optimize Nash Social Welfare (NSW)—a measure of efficiency based on the geometric mean of agents' utilities—we can design local search algorithms that achieve both fairness and a constant-factor approximation of NSW. These algorithms provide a simultaneous guarantee of fairness and efficiency, which is close to optimal for this class of problems.

1.1.2 Online Computation Model

The online fair division setting introduces additional complexity due to the sequential arrival of resources and the need for immediate, irrevocable, allocation decisions [Albers, 1997]. Consider, for instance, the allocation of perishable medical supplies, such as vaccines or organ transplants. Unlike the offline setting—where all resources are known beforehand—here, we must assign resources as they become available, without knowledge of future arrivals.

This uncertainty makes achieving fairness significantly more challenging. Even in a system with just two competing agents, the absence of future information can lead to unbounded inequity (see Figure 1.2). In this thesis, I study how fragile the balance between fairness and efficiency becomes under such constraints by introducing online variants of three well-known fair division problems.

The Santa Claus Problem. The Santa Claus problem seeks to fairly allocate presents among

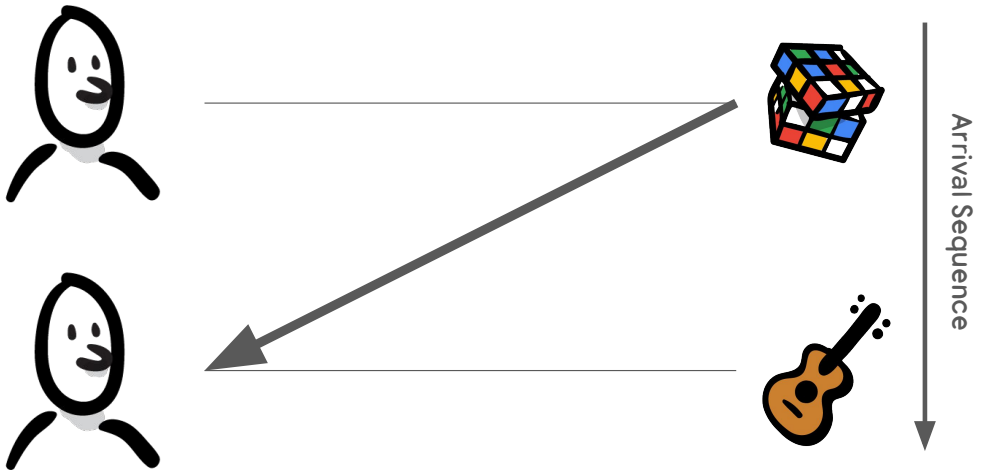


Figure 1.2: Online variant of the allocation example from Figure 1.1. Here the first item arrives and the two agents are indistinguishable, so without loss of generality we allocate to the second. However, upon arrival of the second item, we find that we can no longer allocate the second item.

children, maximizing the minimum satisfaction among all participants [Bansal and Sviridenko, 2006]. Originally studied under a different name in algorithmic game theory, its complexity was first explored through an additive approximation bound and an NP-hardness proof via a reduction from makespan minimization [Lenstra et al., 1990, Bezáková and Dani, 2005]. Subsequent work refined both approximation algorithms and hardness bounds, culminating in a best-known quasi-polynomial $\tilde{O}(n^{\varepsilon})$-approximation, where $\varepsilon = \Omega(\log \log n / \log n)$ [Chakrabarty et al., 2009].

In real-world applications, the set of resources to be allocated is often not known in advance. For instance, in online advertising, ad-space providers receive bids from competing agents in real-time, requiring an immediate allocation decision [Buchbinder et al., 2007, Blum et al., 2006, Hajiaghayi et al., 2007, Zhou et al., 2008]. Motivated by such applications, we study the online Santa Claus problem, where items arrive sequentially and must be assigned immediately.

With mild assumptions on the input instance, we design an algorithm that achieves an ap-

proximation of $(1-\varepsilon)$ to the offline optimum for any $\varepsilon > 0$. Our approach leverages randomized greedy allocation with an essential "forgetting" step, allowing the algorithm to adapt to new arrivals dynamically. These theoretical insights have since informed real-world improvements in food bank distribution systems [Mertzanidis et al., 2024], where perishable goods must be allocated in real-time under fairness constraints.

Matching. The bipartite matching problem is a fundamental challenge in computer science, where nodes in a bipartite graph must be matched to maximize a given objective. In the online setting, studied since the seminal work of Karp, Vazirani, and Vazirani [Karp et al., 1990], one side of the graph (offline nodes) is fixed, while the other (online nodes) arrives sequentially. revealing edges upon arrival. The goal is to maximize the number of matched nodes under immediate decision-making constraints.

The best known deterministic algorithm for online matching achieves a $1/2$-approximation, while the classic randomized algorithm of [Karp et al., 1990] provides a tight $(1-1/e)$-approximation guarantee.

In this thesis, I extend the online matching problem to include fairness constraints, introducing the notion of class envy-freeness (CEF) [Hosseini et al., 2024, Mehta et al., 2013, Hajiaghayi et al., 2024]. This requires that matches be allocated in a way that respects fairness constraints across different agent groups. We develop the first randomized algorithm that guarantees approximate CEF fairness without discarding any online arrivals, thereby resolving an open conjecture in the field.

Our results show that simultaneous fairness and efficiency guarantees are achievable with $O(1)$-approximations to the offline optimum. This relies on a novel proof technique of indepen-

dent interest for other non-additive optimization problems. Additionally, we establish non-trivial upper bounds on the best possible CEF and CPROP approximations under non-wasteful matchings, advancing the theoretical limitations of fair online matching.

A key contribution of this thesis is introducing the concept of the "price of fairness" in online matching. This quantifies the trade-off between achieving maximum matches and ensuring fairness. Specifically, we prove that no online algorithm with an α-CEF guarantee can achieve a matching larger than $\frac{1}{1+\alpha}$ of the optimal. This result provides a rigorous lower bound on the cost of enforcing fairness in real-time decision making scenarios. By characterizing this trade-off, we highlight the fundamental tension between fairness and efficiency in online fair division problems.

1.2 Fair Hierarchical Clustering

Algorithms exhibiting biased outcomes are well-documented even in their early stages of deployment. For instance, algorithmic risk assessment tools used in U.S. courtrooms to predict the likelihood of reoffending have been shown to incorrectly flag Black defendants at twice the rate of white defendants [Angwin et al., 2016b]. Similarly, studies have found that advertisements implying a criminal record appear more frequently when users search for Black-identifying names [Sweeney, 2013]. Bias of this nature has also emerged in healthcare decision-making algorithms [Ledford, 2019], which inadvertently disadvantage marginalized groups. Even when explicitly ignoring attributes such as race, gender, and age, hiring algorithms can develop biases by mimicking patterns learned from biased human decision-makers [Bogen and Rieke, 2018].

These cases represent just a fraction of the known societal harms caused by unfair algo-

rithms, whose influence spans various domains. Mitigating this pervasive issue requires interdisciplinary solutions, potentially combining insights from computer science, ethics, sociology, and law. In this thesis, I focus on a specific facet of the problem: achieving representational equality through partitioning algorithms, particularly in hierarchical clustering.

Our approach to reducing algorithmic bias leverages fair clustering algorithms, which seek to maintain equitable representation of different demographic groups in clustered data. The hierarchical clustering problem involves constructing a multi-level structure of nested clusters from unstructured data (unsupervised learning). Fairness in this context requires that each cluster within the hierarchy adheres to demographic constraints, ensuring proportional representation of various groups at all levels. This prevents the over- or under-representation of any group as clusters are formed and refined.

To illustrate, consider the problem of clustering news articles. Every day, a large number of articles with potentially different political leanings (e.g., right- or left-leaning perspectives) are published. When clustering these articles based on overlapping content, it is crucial to ensure that no particular viewpoint is disproportionately represented in any cluster, regardless of the level of granularity in the hierarchy. Fair hierarchical clustering methods can help achieve this balance by ensuring that clusters reflect diverse viewpoints, without sacrificing structural integrity based on content similarity.

Our work formalizes these goals within a graph-theoretic framework. We model data as a complete graph, where edge weights denote pairwise similarities or differences. A fairness constraint specifies a criterion for determining whether a cluster is fair, and a hierarchical clustering is considered fair if all clusters at every level of the hierarchy meet this constraint. Throughout this thesis, I use the cost function introduced by Dasgupta [Dasgupta, 2016] to measure the

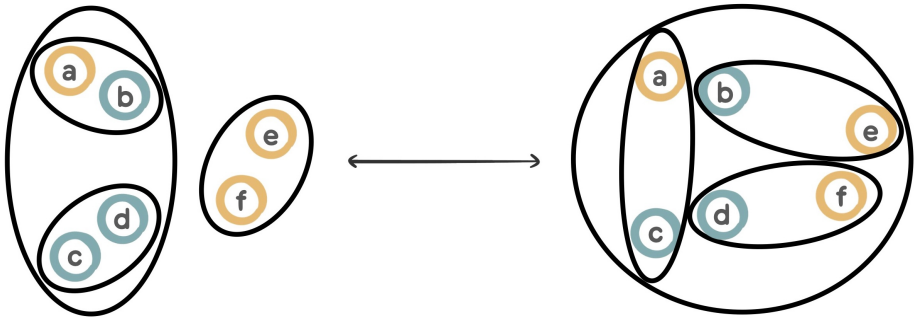


Figure 1.3: Example of an unfair (left) and fair (right) hierarchical clustering for the same dataset. Data are colored to depict their group identification.

quality of clustering outputs under fairness constraints.

Our results accommodate a broad class of fairness definitions that are union-closed—meaning that if two clusters satisfy a fairness constraint, their union does as well. Notably, we handle the generalized bounded representational fairness model [Bera et al., 2019], where each vertex (data point) is assigned a color representing a demographic group. A cluster is considered fair if, for each color i, the proportion of vertices falls within predefined bounds $[\alpha_i, \beta_i]$.

Hierarchical clustering with fairness constraints was first studied by [Ahmadian et al., 2020a], who demonstrated the inherent trade-offs between fairness and solution optimality. Their algorithm achieved an approximation ratio of $O(n^{5/6} \log^{5/4} n)$ for Dasgupta's cost function—only a marginal improvement over the trivial $O(n)$ approximation. In contrast, without fairness constraints, the best-known approximation is $O(\sqrt{\log n})$ [Chierichetti et al., 2017], underscoring the complexity introduced by fairness requirements.

In my initial work on this topic [Knittel et al., 2023b], we developed a combinatorial algorithm that transforms an existing (potentially unfair) clustering hierarchy to improve both fairness and approximate optimality. This algorithm is modular and can be applied to any hierarchical clustering method, enhancing it to produce fair clusters that account for multiple protected

groups. Our approach achieves a near-polylogarithmic approximation ratio of $O(n^{\delta} \log^{5/2} n)$, where δ is a tunable parameter that trades off between accuracy and demographic fairness. Furthermore, this algorithm is the first *explainable* fair hierarchical clustering algorithm, offering interpretability alongside performance improvements.

Building on this foundation, our subsequent work [Knittel et al., 2023a] refined these techniques, achieving a purely polylogarithmic approximation ratio. By optimizing the tree-based representation of clustering hierarchies, we narrowed the gap between the cost of fair clustering and the best unfair solutions. This result demonstrates that fairness can be achieved with only a minor efficiency trade-off, supporting the broader principle of a minimal price of fairness. This insight has important implications for algorithm design, suggesting that fairness constraints can be integrated with minimal impact on overall performance.

In this section of the thesis, I explore these theoretical and algorithmic advancements in greater detail, providing formal analyses and experimental validations. These results contribute to a growing body of research that seeks to reconcile the often competing demands of fairness and efficiency in machine learning and algorithmic systems.

1.3 Roadmap

This dissertation covers the aforementioned works in their entirety. It is divided into three parts. In Part I, we discuss our results in fair division in the *offline* computation model, introducing our model for dividing amongst asymmetric agents (Chapter 2), and extending to a new problem domain with multi-graph instances (Chapter 3). In Part II, we further consider the fair division problem in the *online* computation model, adding considerable complexity to the offline

setting in order to better capture modern compute environments. In Chapter 4, we initiate study of the well-known Santa Claus problem to this online setting. We lastly merge the matching problem with fair division with the first known randomized solutions to the so-called "online class fair matching" problem in Chapter 5. In Part III, we turn our attention to fair hierarchical clustering. My two major works on this problem, namely the first *explainable* algorithm (Chapter 6) and a true polylogarithmic approximation algorithm (Chapter 7). These works provide considerably improvements on the initial study of the problem, effectively resolving the question as to whether fair solutions can exist that are close to the unfair optimum.

Part I

Fair Division in the Offline Model

Chapter 2: Almost Envy-Free Allocation of Indivisible Goods or Chores

2.1 Introduction

The fair allocation of resources is a fundamental problem that has been extensively studied in economics and, more recently, in computer science [Bouveret et al., 2016, Brams and Taylor, 1995, Peterson and Su, 2002, Peterson and Su, 2009, Pikhurko, 2000, Procaccia, 2009, Robertson and Webb, 1998]. Fairness considerations arise in a wide range of applications, including land division, divorce settlements, and the distribution of natural resources.

A well-established criterion for fairness is *envy-freeness*, which ensures that no agent prefers another's allocation over their own. In the case of goods, this means each agent receives a bundle they consider at least as valuable as any other; for chores, it translates to minimizing the burden each agent bears relative to others. However, when resources are indivisible, achieving envy-freeness is often impossible, and computing such an allocation can be computationally challenging. To address these limitations, several relaxations of envy-freeness have been proposed.

In this work, we examine two key relaxations of envy-freeness: *envy-free up to one good* (EF1) and *envy-free up to any good* (EFX). An allocation of indivisible goods satisfies EF1 if any envy an agent has toward another's share can be eliminated by removing a single item from the envied bundle. Polynomial-time algorithms for computing EF1 allocations exist for any number of agents [Lipton et al., 2004, Budish, 2011].

EFX strengthens this criterion by requiring that no agent envies another after the removal of *any* item from the other's allocation. While theoretically more stringent than EF1, the existence of EFX allocations remains an open question despite considerable research efforts. The first *approximate* EFX results were established by [Plaut and Roughgarden, 2020], with subsequent refinements by [Amanatidis et al., 2020] and [Farhadi et al., 2021].

For chore division, the literature is comparatively sparse. The first discrete and bounded envy-free protocol accommodating any number of agents was only introduced in 2018 [Dehghani et al., 2018].

In both the good and chore division problems, the literature noted above is restricted to the case where each agent's perspective is weighted equally. However, more recently, the generalization has been proposed where agents have a valuation (or cost in the case of chores) associated with each item as well as an "entitlement" or weighting [Aziz et al., 2019, Babaioff et al., 2024, Chakraborty et al., 2020, Farhadi et al., 2019]. This problem setting captures significantly more characteristics of real-world fair division issues than that of the equal entitlement case.

To further emphasize this natural problem, consider the recent example that arose during the height of the COVID-19 pandemic. When a vaccine was in production, producers were faced with the question of how to fairly distribute doses in a manner that respects all parties while also mitigating further spread of the virus. In this context, it is not enough to just give each nation an equal share of the supply —we must now take into account population densities and each nation's medical infrastructure among a plethora of other factors. This naturally gives rise to agent weightings that, as we will show in the paper, result in a considerable complication of the fair allocation problem.

	1WEF	WEFX (id)	WEFX (int)	WEFX (add)	XWEF	α-WEFX
$n = 2$	✓ Thm. 2.4.3	✓ Thm. 2.3.2	✓ Thm. 2.3.6	✗ Thm. 2.3.4	✗ Thm. 2.4.1	✓ Thm. 2.3.7
$n = 3$	✓ Thm. 2.4.3	✓ Thm. 2.3.2	?	✗ Thm. 2.3.5	✗ Thm. 2.4.2	?
$n > 3$	✓ Thm. 2.4.3	✓ Thm. 2.3.2	?	?	?	?

Table 2.1: A summary of our existence results. "✓" indicates the type of allocation specified by the column is guaranteed to exist in the setting specified by the row, while "✗" indicates that we give a counterexample and "?" indicates an open question. "id.", "int.", and "add." serve as shorthand for identical valuation, integer valued weights, and additive valuation assumptions respectively.

2.1.1 Our Contributions

Although the existence of EFX allocations is a major open problem in the field, we demonstrate that weighted EFX (WEFX), the generalized version of EFX for agents with entitlements, cannot be guaranteed in general with counterexamples that are at best 0.786-WEFX for $n = 2$ agents (or 0.795 for $n = 3$), giving the first upper bound on the problem. We further provide analogous impossibility results for the chore setting on $n = 2$ or 3 agents.

Nevertheless, we present several positive results under specific assumptions on the problem instance. In particular, we show that WEFX allocations always exist and can be computed efficiently when agents have identical additive or ordinal valuation profiles. For general additive profiles, we modify the classic "I-cut-you-choose" procedure to ensure the WEFX property when dealing with two agents with integer-valued weights. Lastly, we introduce a novel procedure for an approximate WEFX solution under normalized weights (summing to 1), employing an algorithm akin to the extensively studied "moving-knife" technique. This approach attains an approximate factor of $\frac{w}{2\sqrt[3]{m}}$, where w represents the highest agent weighting and m is the number of items. Synthesis of the above results demonstrates that the weighted version of the envy-free

up to any good problem is *considerably* more challenging than its unweighted counterpart, even in the most simplistic case of $n = 2$. These results also nicely address an open direction posed in [Chakraborty et al., 2020] on this notion of weighted EFX allocations.

For the chore division problem, we extend the work of [Chakraborty et al., 2020] to give a weighted-picking sequence procedure for allocating indivisible chores to weighted agents that is *envy-free up to one chore (1WEF)* and runs in polynomial time. This positive result further reaffirms the gap in complexity between our two relaxed notions of envy-freeness. We here note that independent and in parallel to our work, [Wu et al., 2023] also proved such allocations exist and can be computed efficiently using a similar procedure. A summary of our existence and impossibility results is presented in Table 2.1.

2.1.2 Further Related Work

An extensive line of work exists concerning fair division of indivisible items, and for a complete overview we defer the reader to the surveys of [Bouveret et al., 2016] and [Markakis, 2017]. Here, we focus only on a subset of papers that are most relevant to the present work.

Previous work on the fair allocation of indivisible items among *asymmetric* agents has largely focused on fairness notions that do not rely on envy. [Farhadi et al., 2019] introduced a weighted extension of the *maximin share* (MMS), a concept extensively studied in other works [Barman and Krishnamurthy, 2020, Budish, 2011, Feige et al., 2021, Garg et al., 2019]. Similarly, [Aziz et al., 2019] examined the weighted MMS (WMMS) framework for chore division, where chores are treated as negatively valued goods and an agent's weight reflects their proportional share of the overall workload.

[Babaioff et al., 2024] investigated competitive equilibrium in settings where agents have asymmetric budgets. More specifically, they revisited MMS allocations for agents with arbitrary entitlements highlighting that the WMMS formulation proposed by [Farhadi et al., 2019] does not fully capture the intuitive requirement that agents with higher entitlements should receive stronger preferential treatment. To address this, they introduced the *AnyPrice Share* (APS) allocation, defining it in both weighted and unweighted contexts, and provided a $\frac{3}{5}$-approximate solution.

A large body of work has explored fairness in the allocation of divisible items through the notion of *proportionality*, which ensures that each agent receives at least a proportional share of the total value [Barbanel, 1996,Brams and Taylor, 1995,Cseh and Fleiner, 2020,Segal-Halevi and Suksompong, 2020]. In the unweighted setting, EF1 allocations also satisfy *proportionality up to one item* (PROP1), meaning that any shortfall in proportionality can be resolved by removing a single item from another agent's allocation. However, this equivalence does not extend to the weighted setting, where EF1 does not necessarily imply WPROP1 [Chakraborty et al., 2020]. More recently, [Aziz et al., 2020] presented a polynomial-time algorithm for computing allocations that satisfy both Pareto optimality and *weighted proportionality up to one item* (WPROP1) for goods and chores in the presence of asymmetric agent weights.

Most closely related to the work in this chapter is [Chakraborty et al., 2020] which considers the allocation problem for indivisible goods with the generalized notion of weighted agents, providing a polynomial time WEF1 algorithm. We build upon this result and the algorithm for the case of chores, while also providing the first results for both WEFX and XWEF with various existential results for each—answering the question of whether these generalized notions of fair allocations can be computed.

2.2 Preliminaries and Basic Definitions

Fair Allocation Problem. An instance of a *weighted* fair allocation problem consists of a set $\mathcal{N}$ of n agents where each $i \in \mathcal{N}$ has weight $w_i > 0$. Let $\mathcal{M}$ be a set of m goods (or chores) with, potentially heterogenous, valuation functions $v_i : 2^m \to \mathbb{R}_{\geq 0}$ (or cost functions $c_i : 2^m \to \mathbb{R}_{\geq 0}$) that are assumed to be additive unless otherwise noted. An allocation of $\mathcal{M}$ is a partition of the set into n disjoint "bundles", $\mathcal{A} = (\mathcal{A}_1, ..., \mathcal{A}_n)$, such that $\bigcup_{i \in [n]} \mathcal{A}_i = \mathcal{M}$ and $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ for any two $i, j \in [n]$. For readability, we will henceforth let $\mathcal{A} = (\mathcal{A}_1, ..., \mathcal{A}_n)$ denote an allocation of goods and $\mathcal{B} = (\mathcal{B}_1, ..., \mathcal{B}_n)$ chores.

Fairness Criteria. We seek to find an allocation over the set of goods (or chores) that is *envy-free*: given an instance of a fair division problem and an allocation $\mathcal{A}$, an agent i envies agent j if they strictly prefer the set $\mathcal{A}_j$ over their own bundle $\mathcal{A}_i$ up to a scaling by their weight.

Definition 2.2.1 (WEF)**.** An allocation is *weighted envy-free* (WEF) if no agent envies another, i.e. for any pair $i, j \in \mathcal{N}$ of agents we have

$$\frac{v_i(\mathcal{A}_i)}{w_i} \geq \frac{v_i(\mathcal{A}_j)}{w_j} \text{ for goods, or } \frac{c_i(\mathcal{B}_i)}{w_i} \leq \frac{c_i(\mathcal{B}_j)}{w_j} \text{ for chores.}$$

However, envy-freeness is too strong a criteria to meet for both indivisible goods and chores (even in the unweighted setting). As such, we define two relaxations of this notion, namely *weighted envy-freeness up to one good or chore* (WEF1 and 1WEF respectively) and *weighted envy-freeness up to any good or chore* (WEFX and XWEF respectively).

Definition 2.2.2 (WEF1 / 1WEF)**.** An allocation of goods $\mathcal{A}$ (or chores $\mathcal{B}$) is said to be

(*i*) weighted envy-free up to one good (WEF1) if for any pair of agents $i, j \in \mathcal{N}$ if

$$\frac{v_i(\mathcal{A}_i)}{w_i} \geq \min_{a \in \mathcal{A}_j} \frac{v_i(\mathcal{A}_j \setminus \{a\})}{w_j}$$

(*ii*) weighted envy-free up to one chore (1WEF) if for any pair of agents $i, j \in \mathcal{N}$ if

$$\min_{b \in \mathcal{B}_i} \frac{c_i(\mathcal{B}_i \setminus \{b\})}{w_i} \leq \frac{c_i(\mathcal{B}_j)}{w_j}$$

Definition 2.2.3 (WEFX / XWEF)**.** An allocation of goods $\mathcal{A}$ (or chores $\mathcal{B}$) is said to be

(*i*) weighted envy-free up to any good (WEFX) if for any pair of agents $i, j \in \mathcal{N}$ if

$$\frac{v_i(\mathcal{A}_i)}{w_i} \geq \max_{a \in \mathcal{A}_j} \frac{v_i(\mathcal{A}_j \setminus \{a\})}{w_j}$$

(*ii*) weighted envy-free up to any chore (XWEF) if for any pair of agents $i, j \in \mathcal{N}$ if

$$\max_{b \in \mathcal{B}_i} \frac{c_i(\mathcal{B}_i \setminus \{b\})}{w_i} \leq \frac{c_i(\mathcal{B}_j)}{w_j}$$

While the envy-freeness up to one and any item are very closely related, they define relaxed notions of fairness that have a large discrepancy in terms of complexity. The current literature has demonstrated that a WEF1 allocation always exists for agents with additive valuation functions [Chakraborty et al., 2020], a result that we here extend to the chore division setting with a polynomial time algorithm. However, the problems of the existence of WEFX allocations for goods, as well as XWEF for chores, remain open. In this chapter, we show that WEFX allocations are guaranteed to exist under certain, restrictive, problem assumptions. However, in general,

we show that WEFX and XWEF allocations are not guaranteed to exist for agents with additive cost functions, but overcome this inherent barrier with a polynomial time algorithm that yields an *approximate* allocation for two agents.

Definition 2.2.4 (α-WEFX / β-XWEF)**.** For constants $\alpha \leq 1$ and $\beta \geq 1$, an allocation of goods $\mathcal{A}$ (or chores $\mathcal{B}$) is called

(*i*) α-approximate envy-free up to any good (α-WEFX) if for any $i, j \in \mathcal{N}$

$$\frac{v_i(\mathcal{A}_i)}{w_i} \geq \alpha \cdot \max_{a \in \mathcal{A}_j} \frac{v_i(\mathcal{A}_j \setminus \{a\})}{w_j}$$

(*ii*) β-approximate envy-free up to any chore (β-XWEF) if for any $i, j \in \mathcal{N}$

$$\max_{b \in \mathcal{B}_i} \frac{c_i(\mathcal{B}_i \setminus \{b\})}{w_i} \leq \beta \cdot \frac{c_i(\mathcal{B}_j)}{w_j}$$

In the following two sections, we more formally state our theoretical results and provide the intuition for their proofs. All proofs are deferred to Section 2.5.

2.3 Weighted Division of Indivisible Goods

We here investigate the issue of fairly distributing a collection of indivisible goods among weighted agents, considering different problem assumptions. While achieving envy-freeness through a complete allocation is the ideal outcome, it is not always easily computed or assured by an algorithm. Consequently, we often turn to the less strict criteria of envy-freeness up to any item. The overall existence of such criteria remains an unresolved challenge in the realm of fair allocation for both unweighted and weighted systems.

In this context, we present certain assumptions that prove adequate for ensuring the existence of WEFX allocations. Moreover, when these assumptions are relaxed, we exhibit the non-existence of such allocations.

2.3.1 Identical Valuations

To begin, we simplify the scenario by initially considering a situation where the valuation functions are the same across all agents. In this setting, we observe that the classic *envy-cycle elimination* method from [Lipton et al., 2004] produces the desired WEFX allocation with only a minor adjustment. Specifically, the "envy-graph" is modified to have an edge from agent i to agent j if and only if i has *weighted* envy towards j. This gives us the following positive results.

Theorem 2.3.1. *The weighted version of [Lipton et al., 2004]'s envy-cycle elimination algorithm produces a complete WEFX allocation for agents with identical additive valuation functions.*

Moreover, we can slightly generalize this result to encompass the setting in which agents share a preferential ordering over the set of goods.

Theorem 2.3.2. *The weighted version of the envy-cycle elimination algorithm produces a complete WEFX allocation for agents with identical ordinal preferences.*

The simple model of identical valuations however affords results that are considerably more positive than the general case as exhibited by the below result.

Proposition 2.3.3 ([Chakraborty et al., 2020])**.** *The weighted version of the envy-cycle elimination algorithm may not produce a complete WEF1 allocation, even in a problem instance with two agents and additive valuations.*

2.3.2 General Impossibility

For $n = 2$ and 3, it is known that EFX allocations are guaranteed to exist under additive valuation functions [Plaut and Roughgarden, 2020, Chaudhury et al., 2020]. However, we here present the first existential results that show the same result does *not* hold in the weighted setting—a considerable deviation between the two problem contexts.

We first sketch the construction for a constant factor approximation upper bound to the WEFX problem on two agents, presenting a complete analysis of the result in the Section 2.5.1.

Theorem 2.3.4. *For $n = 2$ agents with additive valuation functions and weights w_1 and w_2 such that $w_1 + w_2 = 1$, there exists an instance where a complete WEFX allocation of goods does not exist while a 0.786-WEFX allocation does.*

Proof Sketch. Consider an instance of two agents with weights $w_1 = \alpha$ and $w_2 = 1 - \alpha$, and suppose there are $m = 4$ goods (denoted a_i for $1 \leq i \leq 4$) with the following valuation profiles:

	a_1	a_2	a_3	a_4
v_1	0	1	φ	φ
v_2	0	0	1	1

where $\varphi := (1+\sqrt{5})/2$. We seek to demonstrate that for all the possible allocations of these four items, the largest approximate factor obtainable is 0.786. More specifically, for each allocation we will derive the approximation ratio as a function of the item values and α. After collecting these factors, we will adversarially select α to ensure that all are at most the desired approximate guarantee. □

Furthermore, we expand this straightforward construction to the setting of $n = 3$ agents.

This extension serves to underscore the significant increase in complexity inherent in pursuing the WEFX objective, particularly when contrasted with its unweighted counterpart. The theorem's proof employs the same analytical approach as previously described.

Theorem 2.3.5. *For $n = 3$ agents with additive valuation functions and weights w_1, w_2 and w_3 such that $w_1 + w_2 + w_3 = 1$, there exists an instance where a complete WEFX allocation of goods does not exist while a 0.795-WEFX allocation does.*

As a consequence of these impossibility results, we proceed by designing procedures for the two-agent problem. These procedures yield a WEFX allocation or an approximation to this objective under varied problem assumptions.

2.3.3 Two Agent Procedures

Our first algorithm computes a WEFX allocation for two agents with *integer valued* weights. Formally, we are given two (additive) valuation functions, v_1 and v_2, for a set $\mathcal{M}$ of m items wherein the corresponding agents have weights of 1 and $W \in \mathbb{Z}$. The algorithm involves a modified "I-cut-you-choose" approach, where the agent with the higher weight greedily divides the goods into $W+1$ bundles. Agent 1 then gets the opportunity to select their preferred bundle from among these. The procedure is presented as pseudocode in Algorithm 1.

Theorem 2.3.6. *For $n = 2$ agents with weights 1 and W, along with additive valuation functions, Algorithm 1 computes a WEFX allocation.*

Proof sketch. Given a set of $W+1$ bundles, agent 1's favorite bundle must necessarily be of value at least the average for the partition which necessarily guarantees the WEFX property. The crux of the analysis is thus verifying that, regardless of the first agent's selection, the second agent

Algorithm 1 Integer Weight WEFX Algorithm

Initialize $X_1 \leftarrow \emptyset, X_2 \leftarrow \emptyset$
if $m \leq W$ **then**
    $X_1 \leftarrow \arg\max_{g \in \mathcal{M}} v_1(g)$
    $X_2 \leftarrow \mathcal{M} \setminus X_1$
else
    Initialize $(P_1, ..., P_{W+1}) \leftarrow (\emptyset, ..., \emptyset)$
    while $\mathcal{M} \neq \emptyset$ **do**
        $k = \arg\min_{i \in [W+1]} v_2(P_i)$
        $P_k \leftarrow P_k \cup \{g \in \arg\max_{h \in \mathcal{M}} v_2(h)\}$
    end while
    $X_1 = \arg\max_{k \in [W+1]} v_1(P_k)$
    $X_2 = \mathcal{M} \setminus X_1$
end ifreturn (X_1, X_2)

remains satisfied. By noting that the greedy partitioning procedure conducted by agent 2 reduces to computing an EFX allocation on $n = W + 1$ identical agents, we can verify that the stronger notion of WEFX is necessarily satisfied. □

Despite this positive result, the inherent upper bound demonstrated in Theorem 2.3.4 indicates that adjusting the assumption of agent weightings to be *normalized* instead compels us to design an approximation algorithm. We proceed to present Algorithm 2 which guarantees an *approximate* WEFX allocation for two agents. The algorithm is a modification of the famous moving-knife procedure. Intuitively, the procedure works to minimally satisfy the higher priority agent before reassigning some items to the other agent in an effort to more equitably balance the two and mitigate any envious relationship. This natural algorithm adapted from the unweighted problem setting yields the first approximate WEFX procedure to date. Assuming w is the weight of the higher priority agent, $\alpha = \frac{w}{2\sqrt[3]{m}}$ is our objective approximation factor of WEFX allocation. Our main technical result is stated formally as follows.

Theorem 2.3.7. *Algorithm 2 guarantees an* $\frac{w}{2\sqrt[3]{m}}$*-WEFX allocation for two agents.*

In this algorithm, we first split the items into two bundles $\mathcal{A}_1$ and $\mathcal{A}_2$ based on their normalized valuations for the agents (ie. without loss of generality assume the sum of each agents valuation over all the items is 1). If the first agent has higher value for an item compared to the second agent, the item goes to the first bundle, otherwise it goes to the second bundle. Without loss of generality, we can assume that $v_1(\mathcal{A}_1) \geq w$. Specifically, if $v_1(\mathcal{A}_1) < w$ then we can re-index the two agents so that the inequality holds (since the inequality must hold for the other agent by normalization assumptions on the weights and valuations). We then sort the items in $\mathcal{A}_1$ in descending order based on the valuation function of the first agent, and assume that g_i is the i^{th} highest value item in $\mathcal{A}_1$ for the first agent. Also, assume that k is the maximum number where $v_1(\{g_1, \ldots, g_{k-1}\}) \leq w$ holds. Then, we consider four different allocations as explained in the algorithm, and prove that at least one of these allocations must guarantee α-WEFX.

We proceed to illustrate the result by analyzing each case independently to obtain inequalities that would necessarily hold, and lastly show that all cannot hold at the same time, thus yielding our result. To start, in Case I, we check if we can satisfy the first agent by allocating only one item to her. Hence, the second agent cannot envy this agent after removing the minimum item from $\mathcal{A}_1$. If this does not guarantee an α-WEFX allocation, we can prove the following lemma:

Lemma 2.3.8. *If Case I fails, we have:*

$$1/k < \alpha \left(\frac{1 - w/k}{1 - w} \right) \tag{2.1}$$

Let $v_1(\{g_1, \ldots, g_{k-1}\} = w - \beta$. Lemma 2.3.9 proves that β is bounded by w/k.

Lemma 2.3.9. *If* $v_1(\{g_1, \ldots, g_{k-1}\} = w - \beta$, *then* $0 \leq \beta < w/(k-1)$.

Algorithm 2 Approximate WEFX Algorithm

Input: Two agents with valuation functions v_1 and v_2 with weights w and $1-w$ respectively and set of items $\mathcal{M}$

Output: Allocation $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$
Normalize the valuations, $v_1(\mathcal{M}) = v_2(\mathcal{M}) = 1$
Let $\mathcal{A}_1 = \{g \in \mathcal{M} | v_1(g) \geq v_2(g)\}$, $\mathcal{A}_2 = \mathcal{M} \setminus \mathcal{A}_1$
Sort $\mathcal{A}_1$ in descending order, where g_i be the i-th largest item
Let $k = \arg\max_m v_1(\{g_1, \ldots, g_{m-1}\}) \leq w$
Case I: Set $\mathcal{A}_1 = \{g_1\}$, $\mathcal{A}_2 = \mathcal{M} \setminus \mathcal{A}_1$
if $\mathcal{A}$ is α-WEFX **then**
    return $\mathcal{A}$.
end if
Case II: Let $\mathcal{A}_1 = \{g_1, ..., g_{k-1}\}$, $\mathcal{A}_2 = \mathcal{M} \setminus \mathcal{A}_1$
if $\mathcal{A}$ is α-WEFX **then**
    return $\mathcal{A}$.
end if
Case III: Let $\mathcal{A}_1 = \{g_1, ..., g_k\}$, $\mathcal{A}_2 = \mathcal{M} \setminus \mathcal{A}_1$
if $\mathcal{A}$ is α-WEFX **then**
    return $\mathcal{A}$.
end if
Case IV: Let $\mathcal{A}_2 = \arg\max_{g \in B_1} v_2(g)$, $\mathcal{A}_1 = \mathcal{M} \setminus \mathcal{A}_2$

If Case II occurs, an α-WEFX allocation is guaranteed, and the problem is solved. Otherwise, we can prove the following lemma:

Lemma 2.3.10. *If Case II fails, we have:*

$$\frac{w-\beta}{w} < \alpha\left(\frac{1-w+\beta}{1-w}\right) \tag{2.2}$$

If Case III occurs, an α-WEFX allocation is guaranteed, and the problem is solved. Otherwise, we have the following result:

Lemma 2.3.11. *If Case III fails, we have:*

$$\frac{1-w-w/(k-1)+\beta}{1-w} < \alpha\left(\frac{w-\beta}{w}\right) \tag{2.3}$$

If none of these three cases occur, we must have Case IV. If the allocation is α-WEFX, the problem is solved. Otherwise, the following inequality bound must hold:

Lemma 2.3.12. *If Case IV occurs but the allocation is not α-WEFX, we have:*

$$\frac{w/k}{1-w} < \alpha \left(\frac{1-w/k}{w} \right) \tag{2.4}$$

Collecting the above lemmas and showing that Inequalities (2.1-2.4) cannot hold simultaneously gives the final result.

2.4 Weighted Division of Indivisible Chores

We now turn attention to the problem of chore division. Specifically, we present symmetric upper bound results for envy-freeness up to any chore to those of the goods setting before demonstrating that a variant on the WEF1 algorithm of [Chakraborty et al., 2020] yields weighted envy-freeness up to one chore.

2.4.1 General Impossibility

As is the case for the allocation of goods, we cannot in general guarantee that an allocation which is envy-free up to any chore exists for even small problem instances. By slight modification, the construction used for Theorem 2.3.4 yields the following non-existence result.

Theorem 2.4.1. *For $n = 2$ agents with additive cost functions and weights w_1 and w_2 such that $w_1 + w_2 = 1$, there exists an instance where a complete XWEF allocation of chores does not exist while a 1.272-XWEF allocation does.*

Note that the approximation factor here of 1.272 is equivalent to the reciprocal of the approximation factor for WEFX, as the two problems are nearly symmetric to one another. Lastly, this counterexample construction can be extended to the $n = 3$ setting:

Theorem 2.4.2. *For $n = 3$ agents with additive cost functions and weights w_1, w_2 and w_3 such that $w_1 + w_2 + w_3 = 1$, there exists an instance where a complete XWEF allocation of chores does not exist while a 1.214-XWEF allocation does.*

2.4.2 1WEF Algorithm

We lastly present our polynomial time algorithm for weighted envy-freeness up to one chore (1WEF) to compliment the WEF1 algorithm of [Chakraborty et al., 2020]. To achieve this, we must devise a proper sequential picking protocol that allows agents to pick their most preferred chores in a predetermined ordering in a similar vein to the traditional *round-robin* procedure from the symmetric agent literature [Budish, 2011].

In our general case with unequal weights, we devise a *weight-dependent* picking sequence, in addition to an arbitrary ordering final loop, to yield the desired 1WEF allocation for any number of agents and arbitrary weights. Although the proof of our algorithm is intricate and precise, the algorithm itself is intuitive and computationally efficient.

Theorem 2.4.3. *For any number of agents with additive cost functions and arbitrary positive real weights, there exists an algorithm that computes a 1WEF complete allocation in polynomial time.*

The crux of this theorem relies on the carefully constructed picking-sequence of Algorithm 3 so that each agent receives at least one item of higher cost (in the final loop), and the

Algorithm 3 Chore-Division Round-Robin

Input: $\mathcal{L}, (w_1, w_2, ..., w_n), c_i$ for each $i \in \mathcal{N}$

Output: returns 1WEF allocation for the n agents.
Remaining chores $\widehat{\mathcal{L}} \leftarrow \mathcal{L}$
Bundles $B \leftarrow \emptyset, \forall i \in \mathcal{N}$
$t_i \leftarrow 1, \forall i \in \mathcal{N}$
Allocate chores:
while $|\widehat{\mathcal{L}}| > |\mathcal{N}|$ **do**
    $i^* \leftarrow \arg\min_{i \in \mathcal{N}} \frac{t_i}{w_i}$, break ties lexographically
    $l^* \leftarrow \arg\min_{l \in \widehat{\mathcal{L}}} c_{i^*}(l)$
    $B_{i^*} \leftarrow B_{i^*} \cup \{l^*\}$
    $\widehat{\mathcal{L}} \leftarrow \widehat{\mathcal{L}} \setminus l^*$
    $t_{i^*} \leftarrow t_{i^*} + 1$
end while
Allocate Remaining Chores:
for $i \in \mathcal{N}$ **do**
    $l^* \leftarrow \arg\min_{l \in \widehat{\mathcal{L}}} c_i(l)$
    $B_i \leftarrow B_i \cup \{l^*\}$
    $\widehat{\mathcal{L}} \leftarrow \widehat{\mathcal{L}} \setminus l^*$
end for

remaining items are allocated based on a weight-adjusted picking frequency for each agent. We claim that every agent is 1WEF up to the chore selected in the final for loop at every iteration of the while loop. We begin by presenting two insightful lemmas concerning the algorithm itself.

Lemma 2.4.4. *Consider an agent i chosen by Algorithm 3 to pick a chore at some iteration t, and suppose it is not their first pick. Let t_i and t_j denote the number of times agents i and j picked a chore respectively prior to the current iteration. Then $\frac{t_j}{t_i} \geq \frac{w_j}{w_i}$.*

Lemma 2.4.4 is sufficient for ensuring that agent i remains weighted envy-free up to the final chore allocated at each iteration of the loops execution. We verify this guarantee this fact using the following:

Lemma 2.4.5. *Suppose that, for every iteration t in which agent i picks an item prior to their final pick, the number of times that agents i and j have picked chores (t_i and t_j respectively)*

satisfy $\frac{t_j}{t_i} \geq \frac{w_j}{w_i}$. *Then, in every partial allocation (plus the final chore of the for loop) up to and including* i*'s latest pick, agent j is weighted envy-free up to the item allocated in the final loop of Algorithm 3.*

Using Lemmas 2.4.4 and 2.4.5, we now have all the necessary facts to prove Theorem 2.4.3. Intuitively, we maintain a 1WEF invariant up to the chores that are allocated last, which will necessarily incur the most cost for each agent. This is effectively a reversal of the weighted picking sequence procedure of [Chakraborty et al., 2020] that maintains the invariant of a WEF1 allocation up to the *first* allocated item (of largest value to each agent). Though simplistic and intuitive, this extension is non-trivial and relies on a careful analysis of the initial weighted-picking sequence to ensure that our inductive invariant is maintained.

2.5 Proofs

2.5.1 Indivisible Goods

2.5.1.1 Identical Valuations

Proof of Theorem 2.3.1. Let v denote the valuation function for all agents. By construction of the algorithm [Lipton et al., 2004], the (incomplete) allocation at the end of each iteration is guaranteed to be WEFX as long as we can find an agent, say i, towards whom no other agent has weighted envy at the beginning of the iteration. This agent will pick their favorite among the remaining items, which will also be the *least* valuable item in i's bundle according to all agents due to the identical valuations (and the greedy selection by i). At this step in the allocation procedure, we give the item under consideration to agent i and thus any resulting weighted

envy towards i can be eliminated by removing this item. If there is no unenvied agent, then the weighted envy graph consists of at least one cycle; however, under identical valuations, the envy graph cannot have cycles. To see this, suppose that agents $1, 2, ..., \ell$ form a cycle (in that order) for some $\ell \in [n]$. Since agents have identical valuations, it must be that

$$\frac{v(\mathcal{A}_1)}{w_1} < \frac{v(\mathcal{A}_2)}{w_2} < ... < \frac{v(A_\ell)}{w_\ell} < \frac{v(\mathcal{A}_1)}{w_1},$$

which yields a contradiction. □

Proof of Theorem 2.3.2. We now consider the slightly relaxed model of identical ordinal preferences: there exists some ordering of the set, denoted $\sigma \in S_n$, such that $v_i(\sigma_1) \geq v_i(\sigma_2) \geq ... \geq v(\sigma_m)$ for all agents $i \in [n]$.

Again by the envy-cycle elimination procedure, we always allocate to a source node in the envy-graph who naturally picks their favorite item among the remaining options. Since all agents have an identical ordering of the items, this is the favorite item for all agents and is moreover the least valuable within agent i's bundle. Thus, any new envy incurred can be alleviated by removal of this item which implies WEFX as above.

If there is no unenvied agent (source node in the graph), then there must be a cycle in the envy-graph. Thus, suppose that agents $1, 2, ..., \ell$ form a cycle (in that order) for some $\ell \in [n]$, i.e.

$$\frac{v_i(\mathcal{A}_i)}{w_i} < \frac{v_i(\mathcal{A}_{i+1})}{w_{i+1}} \text{ for } i \text{ under modulo } \ell.$$

Let i' be the agent of minimal weight in this cycle. By nature of the envy cycle, we must then

have that

$$\frac{v_{i'}(\mathcal{A}_{i'})}{w_{i'}} < \frac{v_{i'}(\mathcal{A}_{i'+1})}{w_{i'+1}} \Rightarrow v_{i'}(\mathcal{A}_{i'}) < \frac{w_{i'}}{w_{i'+1}} v_{i'}(\mathcal{A}_{i'+1}) < v_{i'}(\mathcal{A}_{i'+1})$$

and therefore the decycling procedure will strictly improve the allocation according to agent i'. We can thus repeat the process until the minimal weight agent is removed from the cycle. Since there always exists an agent of minimal weight, we can always remove an agent from the cycle until a source node exists in the envy-graph. This concludes the proof of WEFX. □

2.5.1.2 Impossibility Results

Proof of Theorem 2.3.4. First, we consider the simplest case where all goods are allocated to one of the agents, denote the allocation $\mathcal{A}_1 = \{a_1, a_2, a_3, a_4\}$:

$$\frac{v_2(\varnothing)}{1-\alpha} = 0 < \frac{2}{\alpha} = \frac{v_2(\mathcal{A}_1)}{\alpha}$$

Which is by definition an envious relationship. We can also see that it is not WEFX by examining the value of the bundle after removing the good of minimal value and once again checking for envy, as this would guarantee that the removal of some item alleviates the envy if possible:

$$\frac{v_2(\varnothing)}{1-\alpha} = 0 < \frac{2}{\alpha} = \max_{a \in \mathcal{A}_1} \frac{v_2(\mathcal{A}_1 \setminus \{a\})}{\alpha}$$

It is also clear that by the additivity of the valuations,

$$\max_{a \in \mathcal{A}_i} \frac{v_i(\mathcal{A}_j \setminus \{a\})}{w_j} \leq \frac{v_i(\mathcal{A}_j)}{w_j}.$$

Thus we need only check this condition when verifying an allocation is not WEFX.

Converse to the above allocation, consider the case where all the goods are allocated to agent 2, denoting the allocation $\mathcal{A}_2 = \{a_1, a_2, a_3, a_4\}$. We once again see this yields an envious relationship which is not WEFX:

$$\frac{v_1(\varnothing)}{\alpha} = 0 < \frac{2\phi + 1}{1 - \alpha} = \max_{a \in \mathcal{A}_2} \frac{v_1(\mathcal{A}_2 \setminus \{a\})}{1 - \alpha}$$

Now consider the case where one agent is allocated exactly one of the goods. The agent who receives the three goods will always be WEFX relative to the other, as removing the other agents single good will yield a value of zero. Thus, we need only consider the agent who receives exactly one good. We first evaluate the possibilities for agent 1:

(1) $\mathcal{A}_2 = \{a_1, a_2, a_3\}$:

$$\frac{v_1(\{a_4\})}{\alpha} = \frac{\phi}{\alpha} < \frac{\phi + 1}{1 - \alpha} = \max_{a \in \mathcal{A}_2} \frac{v_1(\mathcal{A}_2 \setminus \{a\})}{1 - \alpha} \Rightarrow \frac{\phi(1 - \alpha)}{\alpha(\phi + 1)}\text{-WEFX}$$

(2) $\mathcal{A}_2 = \{a_1, a_3, a_4\}$:

$$\frac{v_1(\{a_2\})}{\alpha} = \frac{1}{\alpha} < \frac{2\phi}{1 - \alpha} = \max_{a \in \mathcal{A}_2} \frac{v_1(\mathcal{A}_2 \setminus \{a\})}{1 - \alpha} \Rightarrow \frac{1 - \alpha}{2\phi\alpha}\text{-WEFX}$$

(3) $\mathcal{A}_2 = \{a_2, a_3, a_4\}$:

$$\frac{v_1(\{a_1\})}{\alpha} = 0 \Rightarrow 0\text{-WEFX}$$

We omit the allocation $\mathcal{A}_2 = \{a_1, a_2, a_4\}$ as this yields an equivalent valuation inequality to (1).

Next we consider the converse case of allocating exactly one good to the second agent.

(4) $\mathcal{A}_1 = \{a_1, a_2, a_3\}$:

$$\frac{v_2(\{a_4\})}{1-\alpha} = \frac{1}{1-\alpha} < \frac{1}{\alpha} = \max_{a \in \mathcal{A}_1} \frac{v_2(\mathcal{A}_1 \setminus \{a\})}{\alpha} \Rightarrow \frac{\alpha}{1-\alpha}\text{-WEFX}$$

(5) $\mathcal{A}_1 = \{a_1, a_3, a_4\}$:

$$\frac{v_2(\{a_2\})}{1-\alpha} = 0 \Rightarrow 0\text{-WEFX}$$

We omit here the cases where agent 2 is allocated only a_1 or a_3 as these have equivalent valuations to the above. Lastly, we consider the case where each agent is allocated exactly two goods.

(6) $\mathcal{A}_1 = \{a_1, a_2\}, \mathcal{A}_2 = \{a_3, a_4\}$:

$$\frac{v_1(\mathcal{A}_1)}{\alpha} = \frac{1}{\alpha} < \frac{\phi}{1-\alpha} = \max_{a \in \mathcal{A}_2} \frac{v_1(\mathcal{A}_2 \setminus \{a\})}{1-\alpha} \frac{1-\alpha}{\alpha \cdot \phi}\text{-WEFX}$$

(7) $\mathcal{A}_1 = \{a_1, a_3\}, \mathcal{A}_2 = \{a_2, a_4\}$:

$$\frac{v_2(\mathcal{A}_2)}{1-\alpha} = \frac{1}{1-\alpha} < \frac{1}{\alpha} = \max_{a \in \mathcal{A}_1} \frac{v_2(\mathcal{A}_1 \setminus \{a\})}{\alpha} \frac{\alpha}{1-\alpha}\text{-WEFX}$$

(8) $\mathcal{A}_1 = \{a_3, a_4\}, \mathcal{A}_2 = \{a_1, a_2\}$:

$$\frac{v_2(\mathcal{A}_2)}{1-\alpha} = 0 \Rightarrow 0\text{-WEFX}$$

Finally, noting that $\mathcal{A}_1 = \{a_2, a_3\}, \{a_1, a_4\}$, and $\{a_2, a_4\}$ all result in the same valuations and are thus omitted. We thus conclude that no WEFX allocation exists for the given scenario since all possible unique allocations to the two agents with associated valuation profiles have been examined with none meeting the defined criteria.

It is also important to examine the maximal c-approximate WEFX allocation achieved in the above counterexample:

$$\max_{\alpha \in (0,1)} \left\{ \frac{\alpha}{1-\alpha}, \frac{1-\alpha}{\alpha \cdot \phi}, \frac{\phi(1-\alpha)}{\alpha(\phi+1)} \right\}$$

We finally obtain the maximal approximate factor by equating the three expressions and solving to get $\alpha = 0.44$. Therefore, our maximal approximate factor is 0.786-WEFX. □

Proof of Theorem 2.3.5. For the given problem instance:

	a_1	a_2	a_3	a_4	a_5
v_1	0	0	0	0	1
v_2	ϕ	ϕ	1	0	0
v_3	1	1	0	0	1

with $w = (\frac{3}{19}, \frac{7}{19}, \frac{9}{19})$ we have that the allocation to the three agents with a highest approximate factor of 0.795 is

$$\mathcal{A}_1 = \{a_5\}, \mathcal{A}_2 = \{a_1\}, \mathcal{A}_3 = \{a_2, a_3, a_4\}$$

□

2.5.1.3 Two Agent Procedures

Proof of Theorem 2.3.6. We first analyze the instance where $m \leq W$. Naturally, since agent 1 is allocated only one item (denoted g), the second is WEFX (up to removal of this item). Therefore, we need only analyze the first agent's perspective. By selection $v_1(g) \geq v_1(h)$ for all $h \in \mathcal{M}$ and, therefore, is also greater than the average value of the set. Thus, we have

$$v_1(g) \geq \frac{v_1(\mathcal{M})}{m} \geq \frac{v_1(\mathcal{M})}{W} \geq \frac{v_1(\mathcal{M} \setminus g)}{W}.$$

and the allocation is WEFX.

We now assume that $W > m$. Let $\{X_i\}_{i=1}^{W+1}$ be the $W+1$ partition of $\mathcal{M}$ as constructed in Algorithm 1. Per the algorithm, the first agent receives their favorite of these $W+1$ bundles (without loss of generality, call this X_1). Therefore, we must again have that the value of this bundle to agent 1 is greater than the average value of all remaining bundles. Thus,

$$\begin{aligned} v_1(X_1) &\geq \frac{1}{W} \sum_{i=2}^{W+1} v_i(X_i) \\ &\geq \frac{1}{W} v_1(\mathcal{M} \setminus X_1) \end{aligned}$$

which gives us that the agent is WEF (and, moreover, WEFX) since $\mathcal{M} \setminus X_1$ is exactly the set of items allocated to the second agent.

The less-trivial proof of WEFX from the second agent's perspective relies on a simple yet crucial lemma concerning the greedy partitioning procedure.

Lemma 2.5.1. *For any pair of subsets, P_i and P_j, created in Algorithm 1 such that $v_2(P_i) < v_2(P_j)$, it must hold that*

$$v_2(P_i) \geq v_2(P_j \setminus g) \text{ for all } g \in P_j.$$

We first demonstrate that this lemma gives us the final WEFX property before proceeding to prove the lemma's correctness. First observe that

$$\frac{v_2(\mathcal{M} \setminus X_1)}{W} \geq \min_k v_2(P_k)$$

Naturally, if $v_2(P_k) \geq v_2(X_1)$ for all such partitions given to the second agent, then we are done. Therefore, assume that $v_2(X_1) > \min_k v_2(P_k)$. By Lemma 2.5.1, this further implies that

$$\min_k v_2(P_k) \geq v_2(X_1 \setminus g)$$

for any $g \in X_1$. Combining with the above inequality gives us the desired WEFX property. □

We now circle back to prove Lemma 2.5.1 and complete the analysis.

Proof of Lemma 2.5.1. First, if $|P_j| = 1$ then the result is vacuously true. Thus, we assume that $|P_j| > 1$ which implies that there is some iteration in the run of Algorithm 1 where this bundle has a lower value to the second agent than P_i. Let t be the last iteration in which the j-th bundle receives an item and use superscript notation to denote the bundle at this time (ie. $P_j^t \cup \{g^t\} = P_j$). This further implies that $P_i^t \geq P_j^t = P_j \setminus g$, and note that by the greedy nature of our algorithm, this g is of minimal value in P_j to the second agent. Therefore, at this iteration the result holds and since no further items are allocated to the j-th bundle, we have the result. □

We now turn our attention to the approximate algorithm for normalized agent weightings. We proceed to sequentially prove the necessary lemmas, before obtaining the final theorem's result.

Proof of Lemma 2.3.8. According to the definition of k, $v_1(\{g_1, \ldots, g_k\}) \geq w$. Since g_1 is the largest item in this set according to v_1, we further have that $v_1(g_1) > w/k$. Therefore $v_1(\mathcal{A}_2) < 1 - w/k$. If $\mathcal{A}$ is α-WEFX, we return this allocation. Otherwise, the first agent envies the second even after multiplying through by the approximation factor, thus we have:

$$\frac{w/k}{w} < \alpha \left(\frac{1 - w/k}{1 - w} \right)$$

Simplification yields (2.1). □

Proof of Lemma 2.3.9. By definition, $\beta \geq 0$ necessarily holds. Assume towards contradiction that $\beta \geq w/(k-1)$. Then,

$$\begin{aligned} v_1(g_k) &= v_1(\{g_1, \ldots, g_k\}) - (w - \beta) \\ &\geq w - w + \beta \\ &\geq w/(k-1) \end{aligned}$$

Since the items of $\mathcal{A}_1$ are sorted, we further have that $v_1(g_i) > w/(k-1)$ for every $1 \leq i < k$. Hence, $v_1(\{g_1, \ldots, g_{k-1}\} > w$, a contradiction. □

Proof of Lemma 2.3.10. In Case II, we allocate the first $k-1$ items of $\mathcal{A}_1$ to the first agent, and we have $v_1(\mathcal{A}_1) = w - \beta$. Since these items come from $\mathcal{A}_1$, by definition we have $v_2(\mathcal{A}_1) \leq w - \beta$.

Hence, we have $v_2(\mathcal{A}_2) \geq 1 - w + \beta$ by normalization assumption on the item values. Therefore, if we normalize the allocations based on the weights w and $1 - w$, the second agent cannot envy the first agent. If $\mathcal{A}$ is α-WEFX, this is the final allocation, otherwise, the first agent envies the second even after adding the approximation factor, and therefore Inequality (2.2) holds. □

Proof of Lemma 2.3.11. In Case III, we allocate the first k items of $\mathcal{A}_1$ to the first agent, and have $v_1(\mathcal{A}_1) > w$. In this case, if we normalize the allocations based on the weights w and $1 - w$, the first agent does not envy the second agent. Furthermore, we have $v_1(\{g_1, \ldots, g_{k-1}\} = w - \beta$, and as we discussed in the proof of Lemma 2.3.9, this implies $v_1(g_k) \leq w/(k-1)$. Hence, we can say that $v_1(\mathcal{A}_1) \leq w - \beta + w/(k-1)$. Since the value of every item in $\mathcal{A}_1$ for the first agent is *at least* its value for the second agent, we can say that $v_2(\mathcal{A}_1) \leq w - \beta + w/(k-1)$ and therefore $v_2(\mathcal{A}_2) \geq 1 - w + \beta - w/(k-1)$.

Since for every item in $\mathcal{A}_1$ the value of the first agent is at least the value of the second agent, and given the fact that $v_1(\mathcal{A}_1)$, after removing the minimum item from $\mathcal{A}_1$ (according to the first agent's perspective), is equal to $w - \beta$, we thus argue that $v_2(\mathcal{A}_1)$ is at most $w - \beta$ after removing the minimum item from $v_1(\mathcal{A}_1)$ (according to the second agent's perspective).

If $\mathcal{A}$ is α-WEFX, this is the final allocation, otherwise, the second agent envies the first even after adding the approximation factor, and as a result, Inequality (2.3) should hold. □

Proof of Lemma 2.3.12. In Case IV, we allocate the item with the highest value among all the items in $\mathcal{A}_1$ to the second agent. Since Case III failed, we have $v_2(\{g_1, \ldots, g_k\}) > w$. Hence, the value of at least one of the items in $\{g_1, \ldots, g_k\}$ is at least w/k for the second agent. Therefore, in the Case IV allocation, $v_2(\mathcal{A}_2) \geq w/k$ and $v_2(\mathcal{A}_1) \leq 1 - w/k$.

If $\mathcal{A}$ is α-WEFX, this is the final allocation, otherwise, the second agent envies the first

agent even after adding the approximation factor, and as a result, Inequality (2.4) should hold. (Since we only allocate one item to the second agent, the first agent does not envy the second after removing the minimum item from $\mathcal{A}_2$.) □

We now have the necessary tools to prove our main theorem which states that Algorithm 2 guarantees an α-WEFX allocation. We show this by proving that Inequalities (2.1-2.4) cannot hold at the same time.

Proof of Theorem 2.3.7. If Algorithm 2 does not guarantee an α-WEFX allocation, then all the Inequalities (2.1 - 2.4) should hold. After simplifying Inequality (2.1), we have

$$1 - w < \alpha(k - w)$$

Since $0 < \alpha < 1$, we necessarily have $k > 1$. Hence, we can safely use $k - 1$ as a denominator in Inequality (2.3). Also, after merging Inequalities (2.2) and (2.3), we have:

$$1 - w - w/(k - 1) + \beta < \alpha^2(1 - w + \beta)$$

Since we have $0 < \alpha < 1$, the following inequality must further hold:

$$1 - w - w/(k - 1) < \alpha^2(1 - w)$$

Given $k \geq 2$, we can use the above inequality and write:

$$1 - w - 2w/k < \alpha^2(1 - w) \tag{2.5}$$

After simplifying Inequality (2.4), we have:

$$(w^2/k)(1/\alpha - 1) + w/k + w < 1 \tag{2.6}$$

We rewrite Inequality 2.5 using 2.6 as follows:

$$(w^2/k)(1/\alpha - 1) + w/k + w - w - 2w/k < \alpha^2(1 - w)$$

After simplifying the above inequality, we have:

$$w^2 < \alpha^3 k(1 - w) + \alpha w(w + 1)$$

Since $0 < w < 1$, we can rewrite the above inequality as follows:

$$w^2 < \alpha^3 k + 2\alpha w$$

If we use $\alpha = \frac{w}{2\sqrt[3]{m}}$, we have:

$$w^2 < \frac{w^3 k}{8m} + \frac{w^2}{\sqrt[3]{m}}$$

Since $0 < w < 1$ and $2 \leq k \leq m$, we have:

$$w^2 < \frac{w^2}{8} + \frac{w^2}{\sqrt[3]{2}}$$

which is a contradiction. Hence, all the Inequalities (2.1 - 2.4) cannot hold at the same time, proving the main result. □

2.5.2 Indivisible Chores

Proof of Theorem 2.4.1. Consider the following instance on 4 chores:

	b_1	b_2	b_3	b_4
c_1	0	0	1	1
c_2	0	1	ϕ	ϕ

We proceed by case analysis in the same manner as above. First, we consider the simplest case where all chores are allocated to one of the agents, denote the allocation $\mathcal{B}_2 = \{b_1, b_2, b_3, b_4\}$:

$$\frac{c_2(\mathcal{B}_2)}{1-\alpha} = \frac{2\phi+1}{1-\alpha} > 0 = \frac{c_2(\varnothing)}{\alpha}$$

Which is by definition an envious relationship. We can also see that it is not XWEF by examining the value of the bundle after removing the good of minimal value and once again checking for envy, as this would guarantee that the removal of some item alleviates the envy if possible:

$$\max_{b\in\mathcal{B}_2} \frac{c_2(\mathcal{B}_2)}{1-\alpha} = \frac{2\phi+1}{1-\alpha} > 0 = \frac{c_2(\varnothing)}{\alpha}$$

It is also clear that by the additivity of the valuations, $\max_{a\in\mathcal{B}_i} \frac{c_i(\mathcal{B}_j\setminus\{b\})}{w_j} \leq \frac{c_i(\mathcal{B}_j)}{w_j}$. Thus we need only check this condition when verifying an allocation is not XWEF.

Converse to the above allocation, consider the case where all the chores are allocated to agent 1, denoting the allocation $\mathcal{B}_1 = \{b_1, b_2, b_3, b_4\}$. We once again see this yields an envious

relationship which is not XWEF:

$$\frac{c_1(\mathcal{B}_1)}{\alpha} = \frac{2}{\alpha} < 0 = \max_{a \in \mathcal{B}_2} \frac{c_1(\mathcal{B}_2 \setminus \{b\})}{1-\alpha}$$

Now consider the case where one agent is allocated exactly one of the goods. The agent who receives the one chore will always be XWEF relative to the other, as removing the other agents single chore will yield a value of zero. Thus, we need only consider the agent who receives exactly three chores. We first evaluate the possibilities for agent 1:

(1) $\mathcal{B}_1 = \{b_1, b_2, b_3\}$:

$$\max_{b \in \mathcal{B}_1} \frac{c_1(\mathcal{B}_1 \setminus \{b\})}{\alpha} = \frac{1}{\alpha} > \frac{1}{1-\alpha} = \frac{c_1(\mathcal{B}_2)}{1-\alpha} \Rightarrow \frac{1-\alpha}{\alpha}\text{-XWEF}$$

(2) $\mathcal{B}_1 = \{b_1, b_3, b_4\}$:

$$\max_{b \in \mathcal{B}_1} \frac{c_1(\mathcal{B}_1 \setminus \{b\})}{\alpha} = \frac{2}{\alpha} > 0 = \frac{c_1(\mathcal{B}_2)}{1-\alpha} \Rightarrow 0\text{-XWEF}$$

We omit here the cases where agent 2 is allocated only b_1 or b_3 as these have equivalent valuations to the above.

(3) $\mathcal{B}_2 = \{b_1, b_2, b_3\}$:

$$\max_{b \in \mathcal{B}_2} \frac{c_2(\mathcal{B}_2 \setminus \{b\})}{1-\alpha} = \frac{1+\phi}{1-\alpha} > \frac{\phi}{\alpha} = \frac{c_2(\mathcal{B}_1)}{\alpha} \Rightarrow \frac{\phi(1+\phi)}{\alpha(1-\alpha)}\text{-XWEF}$$

(4) $\mathcal{B}_2 = \{b_1, b_3, b_4\}$:

$$\max_{b \in \mathcal{B}_2} \frac{c_2(\mathcal{B}_2 \setminus \{b\})}{1-\alpha} = \frac{2\phi}{1-\alpha} > \frac{1}{\alpha} = \frac{c_2(\mathcal{B}_1)}{\alpha} \Rightarrow \frac{2\phi\alpha}{1-\alpha}\text{-XWEF}$$

(5) $\mathcal{B}_2 = \{b_2, b_3, b_4\}$:

$$\max_{b \in \mathcal{B}_2} \frac{c_2(\mathcal{B}_2 \setminus \{b\})}{1-\alpha} = \frac{1+2\phi}{1-\alpha} > 0 = \frac{c_2(\mathcal{B}_1)}{\alpha} \Rightarrow 0\text{-XWEF}$$

We omit here the cases where agent 1 is allocated only b_3 as this is equivalent to (3). Lastly, we consider the case where each agent is allocated exactly two goods.

(6) $\mathcal{B}_1 = \{b_2, b_4\}, \mathcal{B}_2 = \{b_1, b_3\}$:

$$\max_{b \in \mathcal{B}_1} \frac{c_1(\mathcal{B}_1 \setminus \{b\})}{\alpha} = \frac{1}{\alpha} > \frac{1}{1-\alpha} = \frac{c_1(\mathcal{B}_2)}{1-\alpha}\frac{1-\alpha}{\alpha}\text{-XWEF}$$

(7) $\mathcal{B}_1 = \{b_1, b_2\}, \mathcal{B}_2 = \{b_3, b_4\}$:

$$\max_{b \in \mathcal{B}_2} \frac{c_2(\mathcal{B}_2 \setminus \{b\})}{1-\alpha} = \frac{\phi}{1-\alpha} > \frac{1}{\alpha} = \frac{c_2(\mathcal{B}_1)}{\alpha} \Rightarrow \frac{\phi\alpha}{1-\alpha}\text{-XWEF}$$

(8) $\mathcal{B}_1 = \{b_3, b_4\}, \mathcal{B}_2 = \{b_1, b_2\}$:

$$\max_{b \in \mathcal{B}_1} \frac{c_1(\mathcal{B}_1 \setminus \{b\})}{\alpha} = \frac{1}{\alpha} > 0 = \frac{c_1(B_2)}{1-\alpha} \Rightarrow 0\text{-XWEF}$$

Finally, noting that $\mathcal{B}_1 = \{b_1, b_3\}, \{b_2, b_3\}$, and $\{b_1, b_4\}$ all result in the same valuations and are thus omitted. We thus conclude that no XWEF allocation exists for the given scenario since

all possible unique allocations to the two agents with associated valuation profiles have been examined with none meeting the defined criteria.

It is also important to examine the minimal k-approximate XWEF allocation achieved in the above counterexample:

$$\min_{\alpha \in (0,1)} \left\{ \frac{1-\alpha}{\alpha}, \frac{\phi\alpha}{1-\alpha}, \frac{\phi(1+\phi))}{\alpha(1-\alpha)} \right\}$$

We finally obtain the maximal approximate factor by equating the three expressions and solving to get $\alpha = 0.44$. Therefore, our maximal approximate factor is 1.272-XWEF. □

Proof of Theorem 2.4.2. For the given problem instance:

	b_1	b_2	b_3	b_4	b_5
c_1	1	1	ϕ	ϕ	1
c_2	1	0	0	1	1
c_3	ϕ	1	0	ϕ	ϕ

with $w = (\frac{1}{15}, \frac{6}{15}, \frac{8}{15})$ we have that the allocation to the three agents with a smallest approximate factor of 1.214 is

$$\mathcal{B}_1 = \{b_1\}, \mathcal{B}_2 = \{b_5\}, \mathcal{B}_3 = \{b_2, b_3, b_4\}$$

□

To conclude, we prove the correctness of our 1WEF round-robing algorithm.

Proof of Lemma 2.4.4. Since agent i is picked at iteration t, it must be that $i \in \arg\min_{i \in \mathcal{N}} \frac{t_i}{w_i}$, and therefore $\frac{t_i}{w_i} \leq \frac{t_j}{w_j} \rightarrow \frac{t_j}{t_i} \geq \frac{w_j}{w_i}$ as $t_i > 0$. □

Proof of Lemma 2.4.5. Consider any iteration t where agent i has been selected to pick the next chore. Let the number of times i appears in the picking sequence before j's first pick be denoted as τ_0, and further denote the number of times i picks between j's k and $(k+1)$-th pick by τ_k. For $r \in [t_j]$, we have that $\sum_{x=1}^{r} \tau_x$ and r are the number of chores selected by i before j selects their $(r+1)$-st item and number of items j has selected respectively, and thus Lemma 2.4.4 gives us

$$\sum_{x=1}^{r} \tau_x \leq r\phi \text{ for all } r \in [t_j]$$

where we let $\phi := \frac{w_i}{w_j}$.

Additionally, we note that anytime agent i was selected, they picked one of the items they valued least among the remaining chores, including items picked later by agent j. Thus, if we let β_x denote the cost of the chore picked by agent j in their x-th pick and α_y^x be the cost of a chore i picks between j's $(x-1)$ and x-th pick, then for all $y \in \tau_x$:

$$\alpha_y^x \leq \min\{\beta_x, \beta_{x+1}, ..., \beta_{t_j}\}$$

Subsequently, since $\tau_x \geq 0$ for any $x \in [t_j]$, then

$$\sum_{y=1}^{\tau_x} \alpha_y^x \leq \tau_x \cdot \min\{\beta_x, \beta_{x+1}, ..., \beta_{t_j}\} \tag{2.7}$$

We claim that for each $r \in [t_j]$,

$$\sum_{x=1}^{r} \sum_{y=1}^{\tau_x} \alpha_y^x \leq \phi \cdot \left(\beta_{last} + \sum_{x=1}^{r} \beta_x \right) + \left(\sum_{x=1}^{r} \tau_x - r\phi \right) \cdot \min\{\beta_r, ..., \beta_{t_j}\}$$

where β_{last} is the cost associated with agent j's chore allocated in the final loop of Algorithm 3. We proceed in proving the claim is true by induction. For the base case of $r = 1$, we utilize inequality (2.7) to get

$$\begin{aligned}\sum_{y=1}^{\tau_1} \alpha_y^1 &\leq \tau_1 \cdot \min\{\beta_1, \beta_2, ..., \beta_{t_j}\} \\ &= \phi \cdot \min\{\beta_1, \beta_2, ..., \beta_{t_j}\} + (\tau_1 - \phi) \cdot \min\{\beta_1, \beta_2, ..., \beta_{t_j}\} \\ &\leq \phi \cdot (\beta_{last} + \beta_1) + (\tau_1 - \phi) \cdot \min\{\beta_1, \beta_2, ..., \beta_{t_j}\}\end{aligned}$$

where the second from the fact that

$$\beta_{last} + \beta_1 \geq \min\{\beta_1, \beta_2, ..., \beta_{t_j}\}.$$

For the inductive step, we assume the claim holds for $r-1$ and we will prove it for r:

$$\begin{aligned}&\sum_{x=1}^{r}\sum_{y=1}^{\tau_x} \alpha_y^x = \sum_{y=1}^{\tau_r} \alpha_y^r + \sum_{x=1}^{r-1}\sum_{y=1}^{\tau_x} \alpha_y^x \\ &\leq \sum_{y=1}^{\tau_r} \alpha_y^r + \phi \cdot \left(\beta_{last} + \sum_{x=1}^{r-1} \beta_x\right) + \left(\sum_{x=1}^{r-1} \tau_x - (r-1)\phi\right) \cdot \min_{r-1 \leq \sigma \leq t_j}\{\beta_\sigma\} \\ &\leq \tau_r \cdot \min_{r \leq \sigma \leq t_j}\{\beta_\sigma\} + \phi \cdot \left(\beta_{last} + \sum_{x=1}^{r-1} \beta_x\right) + \left(\sum_{x=1}^{r-1} \tau_x - (r-1)\phi\right) \cdot \min_{r-1 \leq \sigma \leq t_j}\{\beta_\sigma\} \\ &\leq \phi \cdot \left(\beta_{last} + \sum_{x=1}^{r-1} \beta_x\right) + \left(\sum_{x=1}^{r} \tau_x - (r-1)\phi\right) \cdot \min_{r \leq \sigma \leq t_j}\{\beta_\sigma\} \\ &\leq \phi \cdot \left(\beta_{last} + \sum_{x=1}^{r} \beta_x\right) + \left(\sum_{x=1}^{r} \tau_x - r\phi\right) \cdot \min_{r \leq \sigma \leq t_j}\{\beta_\sigma\}\end{aligned}$$

Where the first inequality invokes the inductive hypothesis, the second uses (2.7), and the final

two steps come from combining and rearranging the terms within the summations as well as the fact that $\min\{\beta_{r-1}, \beta_r, ..., \beta_{t_j}\}$ is necessarily less than or equal to $\min\{\beta_r, \beta_{r+1}, ..., \beta_{t_j}\}$. Lastly, we can see that if we take $r = t_j$, we can once again invoke 2.7 to obtain

$$\begin{aligned}\sum_{x=1}^{t_j}\sum_{y=1}^{\tau_x}\alpha_y^x &\leq \phi\cdot\left(\beta_{last}+\sum_{x=1}^{t_j}\beta_x\right)+\left(\sum_{x=1}^{t_j}\tau_x - t_j\phi\right)\cdot\beta_{t_j}\\ &\leq \phi\cdot\left(\beta_{last}+\sum_{x=1}^{t_j}\beta_x\right)\end{aligned}$$

where the second inequality holds since the second term is at most 0. Observing that $\sum_{x=1}^{t_j}\sum_{y=1}^{\tau_x}\alpha_y^x$ and $\sum_{x=1}^{t_j}\beta_x$ are the partial allocations at iteration t to agents i and j respectively, we can see that on each iteration of the weighted round-robin protocol, agent i is 1WEF up to the final chore allocated in the second loop (β_{last}) to agent j. □

Proof of Theorem 2.4.3. In the instance that $n > l$, we have that only the for loop of Algorithm 3 runs. Thus each agent is allocated *at most* one chore, thus any pair of agents is 1WEF, and more so XWEF.

Now in the more interesting case of $l > n$, we invoke Lemmas 2.4.4 and 2.4.5. The combination of the two lemmas gives us that on any iteration when agent i is picking a chore, they will remain 1WEF up to the final chore allocated in the second loop. As this property holds until a complete allocation is achieved, we have that the final result is 1WEF.

For time complexity of the algorithm, we note that the while loop runs exactly $l - n$ iterations. Within this loop, identification of the minimal $\frac{t_i}{w_i}$ takes at $O(n)$ time followed by the picking of the minimal cost chore, which takes $O(l)$ time. Since the first loop only runs in the event $l > n$, the loop runs in $O(l^2)$ time. Lastly, we have the for loop which runs in $O(n)$

iterations, each taking $O(1)$ time. Therefore, the algorithm runs in time $O(l^2)$. □

2.6 Conclusions, Limitations & Future Work

Our work highlights the increased complexity of the fair allocation problem on both goods and chores when agents are assumed to be asymmetric. While strong negative existential results on WEFX (and XWEF) for the case of $n = 2$ and 3 showcase a considerable deviation from the unweighted problem setting, our simple adaptation on the "I-cut-you-choose" and moving-knife procedures provide a novel first step on both exact and approximate guarantees for these challenging problems. Furthermore, the presented intricate analysis of our $\frac{w}{2\sqrt[3]{m}}$-WEFX suggests that the current methods for approximate EFX guarantees will not necessarily translate to this generalized context. For example, Proposition 2.3.3 reveals the inadequacy of the envy-cycle procedure for general valuations. Furthermore, our efforts to ensure a 1WEF allocation highlight that the standard round-robin procedure requires specific adjustments to encompass the asymmetric agent configuration. Consequently, the amalgamation of these two approaches, though effective in yielding a $(\varphi - 1)$-EFX allocation [Amanatidis et al., 2021], does not naturally expand to address this extended problem.

We leave the question of the existence of approximate WEFX allocations open as well as the connection between WEFX and the other studied fairness notions for the weighted context for future work. We hope the present work inspires more study on the generalized notion of weighted fair division problems as they will require a plethora of novel techniques to tackle.

Chapter 3: Fair and Efficient Allocations on Multi-Graphs

3.1 Introduction

Fair division of scarce resources is a fundamental problem in many disciplines, including computer science, economics, operations research, and social choice theory. In a classical fair division problem, the goal is to "fairly" allocate a set of goods among a set of agents [Steinhaus, 1948]. This is an age-old problem[1], with numerous contemporary applications, e.g., division of family inheritance [Pratt and Zeckhauser, 1990], divorce settlements [Brams and Taylor, 1996], spectrum allocation [Etkin et al., 2005], air traffic management [Vossen, 2002], course allocation [Budish and Cantillon, 2010] and many more.[2]

In many of these applications, the goods to be allocated may be discrete (or indivisible), leading to an explosion of work on *discrete fair division* (see [Amanatidis et al., 2023] for a comprehensive review). Here, the problem is to fairly allocate a set $\mathcal{M}$ of m indivisible goods among a set $\mathcal{N}$ of n agents. Each agent i is equipped with a valuation function $v_i : 2^{\mathcal{M}} \to \mathbb{R}_{\geq 0}$ which shows how much i values each subset of the goods. An allocation of the goods is a partitioning into n disjoint bundles $\langle X_1, X_2, \ldots, X_n \rangle$ such that for all $i \in \mathcal{N}$, X_i is allocated to agent i. In the case that $X_1 \cup \ldots \cup X_n \subsetneq M$, we call the allocation partial. In this paper,

[1] Such problems find very early historical mentions, for instance, in ancient Greek mythology and the Bible.

[2] See `www.spliddit.org` and `www.fairoutcomes.com` for a more detailed explanation of fair division protocols used in day-to-day problems.

we study the existence and computation of fair allocations that are also (approximately) efficient under arguably the strongest fairness and efficiency notions, namely *envy-free up to any good (EFX)* and the *maximum Nash welfare (MNW)* respectively.

Envy-freeness (EF) is one of the most fundamental fairness notions [Stern and Gamow, 1958]. An allocation X is envy-free if no agent strictly prefers another agent's bundle over their own, i.e., $v_i(x_i) \geq v_i(X_j)$ for all agents i and j. While envy-free allocations always exist when goods are divisible [Foley, 1967, Varian, 1973], they may not exist for indivisible goods. For example, when allocating a single item between two agents who both value it positively. This limitation has led to several relaxations of EF, the strongest of which is *envy-freeness up to any good* (EFX).

Envy-freeness up to any good or **EFX** is the "closest analogue of envy-freeness" in discrete fair division [Caragiannis et al., 2019a]. Under EFX, envy from an agent i to another agent j is allowed; however, any such envy must disappear upon the removal of any good from j's bundle, i.e., for all agents i and j and for all $g \in X_j$, $v_i(X_i) \geq v_i(X_j \setminus \{g\})$. Despite significant effort by many researchers, the existence of EFX allocations in general is an open question to this date. As stated in [Procaccia, 2020], EFX is considered one of the most important open problems in fair division.

Extensive research has been conducted on special cases of EFX and slight relaxations of the concept in an effort to better understand and address the problem. Mainly, the focus has been on restricting the number of agents, restricting the valuation functions of the agents, allowing a subset of goods to remain unallocated (also known as EFX with charity), or obtaining approximate EFX allocations. See Section 3.2 for a detailed discussion on these works.

More recently, [Christodoulou et al., 2023] showed the existence of EFX for graphical

valuations, where a simple undirected graph encodes the instance. Here, the agents correspond to the vertices of the graph, and goods correspond to the edges—we will use vertices and agents, as well as edges and goods interchangeably. A good has non-zero marginal value to an agent i, only if it is incident to i. Note that the original problem of EFX can be modelled by graphs when hyper-edges of arbitrary size and multi-edges of arbitrary size are allowed. [Christodoulou et al., 2023] studied the case where the instance can be modeled by a simple graph and proved that EFX allocations exist in this case, but left open the major question of extensions to the more complex multigraph setting. Motivated by this, we ask:

Question. *Can we extend the setting of [Christodoulou et al., 2023] to multigraphs where a pair of agents may share more than one good that they both value positively? Moreover, can we achieve efficiency together with fairness?*

The answer to the second question is negative for (almost) any efficiency notion unless we relax EFX condition. This is due to a result of [Christodoulou et al., 2023] showing the non-existence of *EFX-orientations* in simple graphs—an EFX allocation where every edge is assigned to one of its endpoints. That is, EFX allocations may force lossy allocations where some goods are assigned to agents who have no value for them. This rules out any efficiency guarantees, including even Pareto optimality (PO).

EFX$^+$: Our fairness notion. To circumvent the above negative result, we consider a slightly weaker (original) notion of EFX called *envy-freeness up to any positively valued good* or *EFX$^+$* [Caragiannis et al., 2019b]. An allocation X is EFX$^+$ if for all agents i and j and all goods $g \in X_j$ such that $v_i(X_j) - v_i(X_j \setminus \{g\}) > 0$, then $v_i(X_i) \geq v_i(X_j \setminus \{g\})$. Interestingly enough, EFX was first introduced under this definition in [Caragiannis et al., 2019b].

MNW: Our efficiency notion. To achieve efficiency, we aim to maximize the *Nash welfare (MNW)*, which is defined as the geometric mean of the agents' utilities. Formally, the Nash welfare of an allocation X, denoted $\text{NW}(X)$, is given by

$$\text{NW}(X) = \left(\prod_{i \in [n]} v_i(X_i) \right)^{1/n}.$$

This measure is one of the most important efficiency notions that are well-known to achieve additional fairness and further efficiency guarantees such as *EF up to some good (EF1)* and PO, respectively [Caragiannis et al., 2019b]. Thus, it seamlessly integrates efficiency with fairness. However, computing maximum Nash welfare (MNW) allocations is NP-hard and further challenging to devise a polynomial-time approximation [Cole and Gkatzelis, 2015, Anari et al., 2016, Anari et al., 2018, Ramezani and Endriss, 2009, Cole et al., 2017]. For any factor $\rho \in [0, 1]$, an allocation X is ρ-MNW if $\text{NW}(X) \geq \rho \cdot \text{MNW}$. The best-known polynomial-time algorithm yields a 1.45^{-1}-approximation, highlighting the intricate nature of the problem [Barman et al., 2018]. Furthermore, the problem is APX-hard [Lee, 2017], setting an inherent limitation on achieving a better-than-constant factor approximation.

Our Results. In this paper, we answer the above two questions positively under the fairness notion of EFX$^+$ and the efficiency notion of (approximate) MNW. Extending the [Christodoulou et al., 2023] setting to multigraphs, where a pair of agents may share any number of positively valued goods, we first design a simple algorithm to find an EFX$^+$ allocation. A high-level idea behind the algorithm is as follows: for any pair of agents i and j, we can run the commonly known "cut-and-choose" protocol on the edges between them. The resulting allocation turns out

to be EFX between agents i and j for their common edges. We show that an extension of this protocol ensures an EFX$^+$orientation for multigraphs. However, the resulting allocation may not be efficient.

Our main contribution is to show the existence of EFX$^+$ allocations with good approximation guarantees for MNW, and provide almost tight lower bounds on the approximation. We design a method to find EFX$^+$ allocations which are $\frac{1}{2}$-MNW. The procedure starts with an MNW allocation, and iteratively adjusts it to get the EFX$^+$ guarantee. We show that the later adjustments do not decrease the MNW by more than $\frac{1}{2}$. We complement this result by showing that no approximation better than $\frac{1}{\sqrt{2}}$-MNW can be guaranteed for EFX$^+$ allocations and no approximation better than $\frac{1}{2}$-MNW can be guaranteed for EFX orientations.

3.2 Related Work

First, we discuss the cases for which the existence of EFX allocations is known.

Small Number of Agents. EFX exists for two agents with general monotone valuations [Plaut and Roughgarden, 2020], and for three agents with additive valuations [Chaudhury et al., 2020]. The latter was extended by [Berger et al., 2022] to three agents with nice-cancelable valuations which is a superclass of additive valuations and then further extended by [Akrami et al., 2023] to the case that two agents have general monotone valuations and one has an MMS-feasible valuation which is a superclass of nice-cancelable valuations.

Special Valuations. In the initial study of [Plaut and Roughgarden, 2020], it was demonstrated that EFX allocations are guaranteed to exist and can also be efficiently computed within polynomial time, provided all agents have an identical ordering of goods based on their values. For

scenarios where each agent is limited to two distinct possible values for each good, [Amanatidis et al., 2021] showcased both the existence and efficient computation of EFX allocations, irrespective of the number of agents involved. Subsequently, [Garg and Murhekar, 2023] expanded upon these findings by establishing that such allocations can be achieved in conjunction with PO.

This paper specifically studies the framework of fair division *on graphs* as defined by [Christodoulou et al., 2023]. This model represents agents as nodes and items as shared non-zero marginal value goods between two agents. Their study first considers the notion *orientations*: an allocation with the additional property that items can only be allocated to one of the two adjacent agents. This can be represented by directing the edges of the graph towards the vertex receiving the edge. Computing an EFX orientation is known to be NP-hard via a reduction from circuit satisfiability and it is further demonstrated that such allocations do not exist in general—a direct result of allowing for the removal of zero valued items in the EFX problem. In this problem setting, EFX allocations which are not orientations can always be computed in polynomial time, but we highlight that considering only one item shared between two agents is highly restrictive and, moreover, the EFX allocation is achieved by identifying agents who do not value a good to allocate which is naturally a wasteful decision with respect to any efficiency measure. As a result, we emphasize that in examining instead the EFX$^+$ problem, deriving fair orientations becomes tractable again, a result we achieve through Algorithm 5 and compliment with upper bound constructions.

Fairness and Efficiency. Analogous to [Caragiannis et al., 2019b], in the case of subadditive valuations, it has been established that any allocation maximizing Nash welfare attains a $1/4$-EF1 property [Wu et al., 2021]. Furthermore, for bi-valued valuations, as well as for dichotomous

valuations, allocations maximizing Nash welfare are confirmed to be EFX [Amanatidis et al., 2021, Babaioff et al., 2021]. Most recently, [Feldman et al., 2024] showed remarkable results on *optimal* tradeoff between approximate-EFX and approximate-MNW for additive and subadditive valuations. Specifically, in both settings, they establish the existence of *partial allocations* that are simultaneously α-EFX and guarantee a $\frac{1}{\alpha+1}$-MNW. In agreement with these findings, we give an algorithm to find a *complete allocation* that is exact EFX$^+$ in conjunction with $1/2$-MNW, and obtain a complimentary upper bound. However, with the relaxed non-zero definition of EFX, we provide an upper bound construction with a more optimistic $\frac{1}{\sqrt{2}}$-approximation to Nash welfare.

EFX with Charity. [Caragiannis et al., 2019a] initiated this line of work by proving the existence of partial allocations with $1/2$-MNW guarantee. [Chaudhury et al., 2021b] proved the existence of EFX allocations with at most $n-1$ goods remaining unallocated with the property that no agent envies the set of unallocated goods. The latter was extended to at most $n-2$ unallocated goods by [Berger et al., 2022, Mahara, 2021].

Approximate EFX. Due to the significant challenge of computing complete EFX allocations, a considerable number of papers have focused on the efficient computation of *approximate* EFX allocations which satisfy the desired pairwise guarantee up to a multiplicative factor α. [Plaut and Roughgarden, 2020] gave the first such algorithm which admitted a $1/2$-approximation in pseudopolynomial time for subadditive valuations, with an extension to polynomial time due to [Chan et al., 2019]. The approximation ratio for the additive case was further improved by [Amanatidis et al., 2021, Farhadi et al., 2021] to $\varphi - 1 \approx 0.618$ by combining a round-robin and envy-cycle elimination procedure.

The mixture of these cases has also been studied. For example the existence of partial EFX

allocations when agents have special valuation functions [Akrami et al., 2022] or approximate EFX allocations with charity [Akrami et al., 2023, Chaudhury et al., 2021a, Berendsohn et al., 2022, Jahan et al., 2023].

Alongside EFX, many other fairness notions have been studied. Envy-freeness up to one good (EF1) [Lipton et al., 2004] is a weaker notion of fairness compared to EFX. In EF1, any envy from agent i to agent j must disappear upon the removal of *some* good in j's bundle. EF1 allocations exist when agents have general monotone valuations [Lipton et al., 2004].

For indivisible goods, [Budish, 2011] studied the maximin share (MMS) fairness metric, where we hope to ensure each agent receives a bundle of value at least as much as the maximum value this agent could guarantee for herself if partitioning into n disjoint bundles and keeping the worst of them. A reduction from the PARTITION problem demonstrates that computing an MMS allocation is NP-hard [Woeginger, 1997], but a long line of approximate results exist. For additive valuations, [Kurokawa et al., 2018] show that a $\frac{2}{3}$-MMS allocation always exists and can be computed in polynomial time. Further improvements to a $\frac{3}{4}$-MMS allocation were subsueqently derived [Ghodsi et al., 2018], and later found to be computable in polynomial time [Garg and Taki, 2020]. Recently, [Akrami and Garg, 2024] proved the existence of $(\frac{3}{4} + \frac{3}{3836})$-MMS allocations. When restricting the number of agents or valuation functions stronger results can be guaranteed [Amanatidis et al., 2017, Bouveret and Lemaître, 2016, Gourvès and Monnot, 2019].

[Conitzer et al., 2017] defined the notion of proportionality up to one good (PROP1) wherein each agent must obtain her proportional share if given at most one extra item. Allocations which are PROP1 and PO always exist, and can be computed in polynomial time [Barman and Krishnamurthy, 2019]. A comparable extension on our EFX$^+$ notion is proportionality up to at

most one *positively valued* good (PROPX), the latter of which cannot always be guaranteed [Aziz et al., 2020].

Apart from the discrete allocation of goods, there exists a substantial body of research exploring analogous questions in situations where items are viewed as chores (which are perceived negatively by the individuals) [Dehghani et al., 2018, Springer et al., 2024] or a combination of both desirable goods and undesirable chores, known as mixed manna [Aziz et al., 2022, Aziz et al., 2017, Li et al., 2021, Sun et al., 2021]. Though seemingly symmetric in nature, the study of chore division is less developed and the results from one to the other may not necessarily carry over [Guo et al., 2023].

For more elaborate discussion on the various fairness metrics and corresponding results, we defer the reader to [Amanatidis et al., 2022].

3.3 Preliminaries

A discrete fair division instance consists of n agents and m indivisible goods (items). We consider an extension of graphical valuations [Christodoulou et al., 2023], where an instance is represented by a multigraph, say $G = (V, E)$, with vertices in V corresponding to agents and edges in E corresponding to goods ($|V| = n$ and $|E| = m$). For all $i \in [n]$, let $v_i : 2^E \to \mathbb{R}^+$ be such that $v_i(X)$ is the value of agent i for the subset of items $X \subseteq E$. We assume for all agents i, all edges e not incident to i, and all $S \subseteq E$, $v_i(S \cup \{e\}) = v_i(S)$. Moreover, we restrict attention to the case of additive valuation functions:

$$v_i(S \cup T) = v_i(S) + v_i(T) \text{ for any disjoint } S, T \subseteq E$$

A complete allocation is a partition of E into n partitions denoted as X_i where the i-th agent receives their corresponding bundle. For an allocation $X = \langle X_1, ..., X_n \rangle$, agent i is said to envy agent j if $v_i(X_i) < v_i(X_j)$. An allocation is said to be *envy-free (EF)* if no agent envies another.

Fairness notion. In this work, we consider the strongest relaxation of EF for discrete items, defined as follows:

Definition 3.3.1 (EFX)**.** An allocation X is said to be **envy-free up to any good (EFX)** if, for any agents $i, j \in [n]$, we have:

$$v_i(X_i) \geq v_i(X_k \setminus \{g\}) \quad \forall g \in X_j.$$

As discussed in Section 3.1, it is impossible to achieve any efficiency guarantee together with EFX even for graphical valuations. Therefore, we consider a slight relaxation of EFX, defined as follows, which in fact is the original definition [Caragiannis et al., 2019b].

Definition 3.3.2 (EFX$^+$)**.** An allocation X is said to be **envy-free up to any positive good (EFX$^+$)** if, for any agents $i, j \in [n]$, we have that no agent strongly envies another:

$$v_i(X_i) \geq v_i(X_k \setminus \{g\}) \quad \forall g \in X_j \text{ such that } v_i(X_k \setminus \{g\}) < v_i(X_k).$$

Efficiency notion. We lastly provide a formal definition of our measure of efficiency, the Nash social welfare of an allocation.

Definition 3.3.3 (NW)**.** The **Nash welfare (NW)** of an allocation X is the geometric mean of

Algorithm 4 [Plaut and Roughgarden, 2020] Two Agent EFX

Require: $\{g_i\}_{i=1}^m, (v_1, v_2)$
Ensure: Allocation $X = \langle X_1, X_2 \rangle$
1: $\langle X_1, X_2 \rangle \leftarrow$ LEXIMIN $++(2, m, v_1)$
2: **if** $v_2(X_1) > v_2(X_2)$ **then**
3: **Return** $\langle X_2, X_1 \rangle$
4: **else**
5: **Return** $\langle X_1, X_2 \rangle$
6: **end if**

the n agent valuations for their respective bundles:

$$\text{NW}(X) = \left(\prod_{i \in [n]} v_i(X_i) \right)^{1/n}$$

An allocation X is said to achieve max Nash welfare (MNW) if $\text{NW}(X) = \max_{X'} \text{NW}(X')$, and is to achieve ρ-MNWfor $\rho \in [0, 1]$, if $\text{NW}(X) \geq \rho \cdot \max_{X'} \text{NW}(X')$. We note that allocations which maximize NW are EF1+PO. Moreover, any approximation to MNW implies approximate PO [Caragiannis et al., 2019b].

3.4 EFX$^+$ and Approximate Efficiency

In this section, we give an algorithm to obtain EFX$^+$ allocations which are $\frac{1}{2}$-MNW. As a subroutine, we use the algorithm in [Plaut and Roughgarden, 2020] which achieves EFX allocations for agents with identical valuations. First we give a brief overview of how this algorithm works, and present pseudocode in Algorithm 4.

The procedure is an example of the classical "I-cut-you-choose" allocation procedure. Though the algorithm works for a set of n agents with identical valuations for the set of items to be allocated, we here focus on how it is leveraged in an asymmetric two agent instance to yield an

EFX allocation. Conceptually, this involves allowing the first agent (the cutting agent) to divide the items into two sets, after which the second agent (the choosing agent) selects her preferred set. The remaining set is then assigned to the first agent. Consequently, the optimal strategy for the first agent is to maximize the value of the smaller set so as to be satisfied regardless of the second agent's decision:

$$\langle X_1, X_2 \rangle = \arg \max_{(S_1,S_2)\in\Pi_2(M)} \min\{v_1(S_1), v_1(S_2)\}$$

where $\Pi_2(M)$ is the set of all partitions of M into two disjoint bundles. [Plaut and Roughgarden, 2020] demonstrated that (X_1, X_2) consistently achieves maximin share (MMS) and EFX for the first agent when breaking ties by maximizing the size of the smaller bundle in (X_1, X_2), a scheme referred to as the LEXIMIN++ allocation. As the second agent secures her preferred bundle in (X_1, X_2), the allocation remains envy-free for her. We present the full theorem statement here for clarity.

Theorem 3.4.1 ([Plaut and Roughgarden, 2020]). *Given a set n of agents with identical monotone valuations functions v, and an allocation X of a set of goods M to these agents, the output of PR(v, X) is an allocation Y with the following properties:*

1. *Y is an EFX allocation, and*
2. $\min_{i\in[n]} v(Y_i) \geq \min_{i\in[n]} v(X_i)$.

Moreover, if X is not an EFX allocation, the last inequality is strict.

We now give intuition for the proof that Algorithm 5 returns an EFX$^+$ and $\frac{1}{2}$-MNW allocation, with full analysis deferred to Section 3.6. In Algorithm 5, we start with an allocation

Algorithm 5 EFX$^+$ + $\frac{1}{2}$-MNW

Require: $(G([n], E), (v_1, v_2, \ldots, v_n))$
Ensure: Allocation $X = \langle X_1, \ldots, X_n \rangle$
Let X be an allocation with maximum NW
for $(i, j) \in [n]^2$ **do**
  if there exists a good $e \in X_j^{(i,j)}$ such that $v_i(X_i^{(i,j)}) < v_i(X_j^{(i,j)} \setminus \{e\})$ **then**
    $(A, B) \leftarrow \text{PR}(v_i, (X_i^{(i,j)}, X_j^{(i,j)})$
    if $v_j(A) > v_j(B)$ **then**
      $A \leftrightarrow B$
    end if
    $X_i \leftarrow X_i \setminus X_i^{(i,j)} \cup A$
    $X_j \leftarrow X_j \setminus X_j^{(i,j)} \cup B$
  end if
end for
Return $X = \langle X_1, \ldots, X_n \rangle$

that is MNW. Therefore, it is without loss of generality to assume all items between i and j are allocated to i and j. For any pair of agents (i, j) we restrict our attention to the edges in between them, denoted $E^{(i,j)}$. Let $X_i^{(i,j)}$ be the set of edges in $E^{(i,j)}$ that are allocated to i under X. When restricting the set of items only to $E^{(i,j)}$, if i strongly envies j, we let i "divide" these items into two fair bundles from her point of view and ask j to "choose" her favorite bundle. This two agent procedure, restricted to the set of items which i and j share an interest in, leads directly to the EFX$^+$property.

Lemma 3.4.2. *Algorithm 5 returns an EFX$^+$ allocation.*

Note, that the EFX$^+$, rather than EFX, is crucial here as we can only guarantee alleviated envy when restricting attention to $E^{(i,j)}$.

Now, to demonstrate that this cutting procedure to obtain a fair allocation does not diminish our maximum NW too much, we first have to bound the number of item reassignments.

Lemma 3.4.3. *For all $(i, j) \in [n]^2$, Algorithm 5 reallocate the goods in $E^{(i,j)}$ at most once.*

Leveraging this bound, we can subsequently show that each agent's value for their subset of $E^{(i,j)}$ is either increased or diminished by a factor of at most $\frac{1}{2}$. Combining this fact over all such reallocations, we obtain the following lemma.

Lemma 3.4.4. *Algorithm 5 returns a $\frac{1}{2}$-MNW allocation.*

Lastly, we obtain the main theorem's result.

Theorem 3.4.5. *For any multigraph G and additive valuation functions $v_1, \ldots, v_n$, Algorithm 5 returns an allocation Y that is* EFX$^+$ *and $\frac{1}{2}$-MNW.*

Proof. This follows directly from the synthesis of Lemmas 3.4.2 and 3.4.4. □

We note that, crucially, our algorithm for reallocating items to alleviate envy does not in fact rely on the initial allocation being one maximizing Nash welfare. As a result, even starting from a suboptimal allocation such as the resultant of the 1.4^{-1}-NW from [Barman et al., 2018], we can maintain strong approximate efficiency guarantees. Concretely, for any initial ρ-MNW allocation, our procedure yields an EFX$^+$ algorithm which is $\frac{\rho}{2}$-MNW.

3.5 Upper Bounds on Approximate MNW

In this section, we give upper bounds on approximation to the maximum Nash welfare when the allocation is EFX$^+$ or EFX. Namely, we sketch the adversarial input constructions that yield Theorem 3.5.1 and 3.5.2, with full proofs deferred to Section 3.6.

To bound the the efficiency guarantees, we construct an identical instance for every pair of agents which cannot maximize NW while maintaining EFX$^+$. Concretely, consider the following instance between two agents with $m = 3$ items to be allocated.:

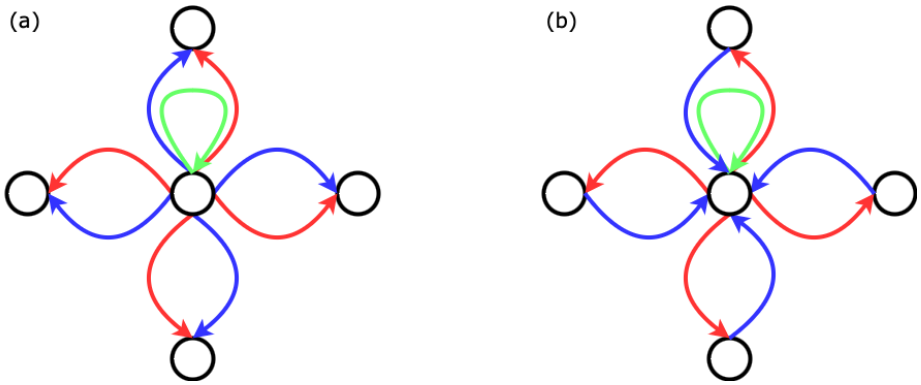


Figure 3.1: Allocation problem where any EFX allocation is at most $2^{\frac{1-n}{n}}$-NW maximizing depicted with $n = 5$. The central node is the first agent and the outer represent the remaining symmetric agents. Red edges are of value 1 to both agents, blue edges are value $1 - \varepsilon$ and the green edge is value $1 - \varepsilon$ to the first agent. Arrows indicate allocation to an agent and (a) depicts the Nash welfare maximizing allocation whereas (b) depicts the EFX.

	g_1	g_2	g_3
v_1	$1+\varepsilon$	1	ε
v_2	$1+\varepsilon$	ε	1

Without loss of generality, we allocate g_1 to the first agent. In order to maximize the Nash welfare, we must then allocate g_2 and g_3 to the first and second agents respectively, violating EFX$^+$.

Theorem 3.5.1. *For any $\delta > \frac{1}{\sqrt{2}}$, δ-MNW and EFX$^+$ allocations are incompatible in the multigraph setting: there exists a multigraph instance in which no EFX$^+$ allocation is δ-MNW.*

For the stronger notion of EFX orientations, we must now construct an instance that exploits the non-locality of EFX (as compared to EFX$^+$). Specifically, we devise an instance with one central agent that shares an two items of interest with all other $n - 1$ periphery agents.

Formally, the instance is as follows:

$$v_1(g_j) = \begin{cases} 1-\varepsilon \text{ for } j = 1 \\ 1 \text{ for } 1 < j \leq n \\ \frac{\varepsilon}{2n} \text{ otherwise} \end{cases}, v_i(g_j) = \begin{cases} 1 \text{ for } j = i \\ 1-\varepsilon \text{ for } j = 2n - i \end{cases}$$

where $i = 1$ is our central agent.

For such an input, NW is maximized when allocating the central agent only one item, whereas the EFX allocation evenly divides the items with periphery agents. This discrepancy between the two allocations which yields a slightly weaker bound of $\frac{1}{2}$ on the maximal NW is depicted for five agents in Figure 3.1.

Theorem 3.5.2. *For any $\delta > \frac{1}{2}$, δ-MNW and EFX orientations are incompatible: there exists a multigraph instance with an EFX orientation in which no EFX orientation is δ-MNW.*

We highlight that this result agrees with the major finding of [Feldman et al., 2024] which guarantees that any complete EFX allocation for additive valuation functions must be at most $\frac{1}{2}$-NW maximizing. The result is achieved by iteratively dropping items from a maximal NW allocation to retain the EFX guarantee, and we note that our construction for the multi-graph problem can be extended to one on general additive valuation instances to match. The construc-

tion is as follows:

$$v_i(g_j) = \begin{cases} 1 \text{ for } j < n \\ 1 - \varepsilon \text{ for } j = n + i \\ \frac{\varepsilon}{2n} \text{ otherwise} \end{cases}$$

and it is straightforward to show that the maximal Nash welfare is $\sqrt[n]{(2-\varepsilon)^{n-1}(1-\varepsilon)}$ whereas any EFX allocation will have a geometric mean of at most 1.

3.6 Proofs

3.6.1 EFX$^+$Guarantees

Proof of 3.4.2. Consider any two arbitrary agents i and j. We prove i does not strongly envy j after the end of the algorithm. Let X be the allocation after the iteration of the for-loop in which (i, j) is realized. By Theorem 3.4.1, for all $g \in X_j^{(i,j)}$, we have

$$v_i(X_i^{(i,j)}) \geq v_i(X_j^{(i,j)} \setminus \{g\}). \tag{3.1}$$

After this step, the allocation of the edges between i and j can only change when (j, i) is realized in the for-loop. However, in that case i is the choosing agent and thus Inequality (3.1) still holds. Now let Y be the output of the Algorithm. We have $X_i^{(i,j)} = Y_i^{(i,j)} \subseteq Y_i$ and $X_j^{(i,j)} = Y_j^{(i,j)} \subseteq Y_j$.

Therefore, for all $g \in Y_j^{(i,j)}$,

$$v_i(Y_i) \geq v_i(X_i^{(i,j)}) \geq v_i(X_j^{(i,j)} \setminus \{g\}) = v_i(Y_j \setminus \{g\}).$$

Note that for all $g \in Y_j$ such that $v_i(Y_j \setminus \{g\}) < v_i(Y_j)$, we have $g \in Y_j^{(i,j)}$. Hence, we proved i does not strongly envy j. □

Proof of Lemma 3.4.3. If Algorithm 5 does not reallocate the goods in $E^{(i,j)}$, then the lemma follows. Otherwise, let us assume $E^{(i,j)}$ gets reallocated for the first time when (i, j) is realized in the for-loop. Thus, agent i divides and agent j chooses. Let X be the allocation after this step. We have, $v_j(X_j^{(i,j)}) \geq v_j(X_i^{(i,j)})$. After this step, the allocation of the edges between i and j can only change when (j, i) is realized in the for-loop. However, since $v_j(X_j^{(i,j)}) \geq v_j(X_i^{(i,j)})$, the allocation remains the same in this step. □

Proof of Lemma 3.4.4. Let X be an MNW allocation that Algorithm 5 starts with and let Y be the output of the algorithm. Fix an arbitrary agent i. For all other agents j, first we prove $v_i(Y_i^{(i,j)}) \geq \frac{1}{2} \cdot v_i(X_i^{(i,j)})$.

1. If $Y_i^{(i,j)} = X_i^{(i,j)}$ then the claim follows.

Otherwise, by Lemma 3.4.3, $E^{(i,j)}$ was reallocated only once. In this reallocation, i was either the cutting agent or the choosing agent.

2. If i was the cutting agent, by Theorem 3.4.1, $v_i(Y_i^{(i,j)}) > v_i(X_i^{(i,j)})$.

3. If i was the choosing agent, then $v_i(Y_i^{(i,j)}) \geq v_i(Y_j^{(i,j)})$. We have,

$$
\begin{aligned}
v_i(Y_i^{(i,j)}) &\geq \frac{1}{2} \cdot v_i(E^{(i,j)}) && (v_i(E^{(i,j)}) = v_i(Y_i^{(i,j)}) + v_i(Y_j^{(i,j)})) \\
&\geq \frac{1}{2} \cdot v_i(X_i^{(i,j)}). && (X_i^{(i,j)} \subseteq E^{(i,j)})
\end{aligned}
$$

Thus, we have

$$
\begin{aligned}
v_i(Y_i) &= \sum_{j \neq i} v_i\left(Y_i^{(i,j)}\right) \\
&\geq \sum_{j \neq i} \frac{1}{2} \cdot v_i\left(X_i^{(i,j)}\right) \\
&= \frac{1}{2} \cdot v_i(X_i).
\end{aligned}
$$

Now by direct computation we have that

$$
\begin{aligned}
\text{NW}(Y) &= \left(\prod_{i \in [n]} v_i(Y_i)\right)^{1/n} \\
&\geq \left(\prod_{i \in [n]} \frac{1}{2} \cdot v_i(X_i)\right)^{1/n} \\
&= \frac{1}{2}\left(\prod_{i \in [n]} v_i(X_i)\right)^{1/n}.
\end{aligned}
$$

□

3.6.2 MNW Upper Bounds

Proof of Theorem 3.5.1. Consider an instance with $n = 2$ and $m = 3$ with the following valuation profiles:

	g_1	g_2	g_3
v_1	$1+\varepsilon$	1	ε
v_2	$1+\varepsilon$	ε	1

where $\delta - \frac{1}{\sqrt{2}} > \varepsilon > 0$. Without loss of generality, we allocate g_1 to the first agent. In order to maximize the Nash welfare, we must then allocate g_2 and g_3 to the first and second agents respectively. This gives a maximal geometric mean of

$$\text{NW}(X) = \sqrt{v_1(X_1) \cdot v_2(X_2)} = \sqrt{2+\varepsilon}$$

and the allocation is not EFX$^+$ since the removal of g_2 does not alleviate the envy of the second agent:

$$v_2(X_2) = 1 < 1 + \varepsilon = v_2(X_1 \setminus g_2).$$

Observe that if we allocate g_1 to the first agent, then the only EFX$^+$ allocation would set $X_2 = \{g_2, g_3\}$. Thus, an EFX$^+$ allocation for the given input has a geometric mean of at most

$$\text{NW}(Y) = \sqrt{v_1(X_1) \cdot v_2(X_2)} = 1 + \varepsilon.$$

Therefore, we have that the approximation ratio is

$$\frac{\text{NW}(Y)}{\text{NW}(X)} = \frac{1+\varepsilon}{\sqrt{2+\varepsilon}} < \frac{1}{\sqrt{2}} + \varepsilon < \delta$$

completing the result. Note that by copying this pair of agents, we can get instances of arbitrary large size with the same lower bound on approximate MNW for EFX$^+$ allocations. □

Proof of Theorem 3.5.2. Consider the following instance for n agents with $m = 2n - 1$ items defined by the given valuation profiles for some small $\varepsilon > 0$ (graphically depicted in Figure 3.1):

$$v_1(g_j) = \begin{cases} 1 - \varepsilon \text{ for } j = 1 \\ 1 \text{ for } 1 < j \leq n \\ \frac{\varepsilon}{2n} \text{ otherwise} \end{cases}, v_i(g_j) = \begin{cases} 1 \text{ for } j = i \\ 1 - \varepsilon \text{ for } j = 2n - i \end{cases}$$

The allocation $X^* = \langle X_1^*, \ldots, X_n^* \rangle$ where $X_1^* = \{g_1\}$ and for $i \neq 1$ we have $X_i^* = \{g_i, g_{2n-i}\}$ maximizes the geometric mean to obtain:

$$\text{NW}(X^*) = \sqrt[n]{(2 - \varepsilon)^{n-1}(1 - \varepsilon)}$$

However, for this allocation, we have that the first agent strongly envies all other agents:

$$v_1(X_1) = 1 - \varepsilon < 1 = \max_{g \in X_i} v_i(X_i \setminus g)$$

for any $i \neq 1$. Now consider any EFX allocation $X = \langle X_1, ..., X_n \rangle$. If any agent $i > 1$ does not receive their one valued good, then they must envy the first agent:

$$v_i(X_i) \leq 1 - \varepsilon < 1 = \max_{g \in X_1} v_i(X_1 \setminus g)$$

Thus, any EFX allocation must give the one valued items to their respective agents. Moreover, assuming this allocation, then the first agent must receive all other items in order not to envy the

others. This yields a Nash welfare of $\text{NW}(X) = \sqrt[n]{1 - \frac{\varepsilon}{n}} < 1$. Lastly, observe that

$$\begin{aligned}
\text{NW}(X^*) &= \sqrt[n]{(2-\varepsilon)^{n-1}(1-\varepsilon)} \\
&= 2^{1-\frac{1}{n}} \left(1 - \frac{\varepsilon}{2}\right)^{\frac{n-1}{n}} (1-\varepsilon)^{\frac{1}{n}} \\
&> 2^{1-\frac{1}{n}} \left(1 - \frac{\varepsilon}{2} \cdot \frac{n-1}{n}\right) \left(1 - \frac{\varepsilon}{n}\right) \\
&> 2^{1-\frac{1}{n}} - 2\varepsilon,
\end{aligned}$$

where the first inequality is due to $(1-x)^r > 1 - rx$ for $r > 0$. Thus, we have the result. □

3.7 Conclusions, Limitations & Future Work

In this paper we extend the novel perspective of [Christodoulou et al., 2023] on discrete fair division to the richer class of multigraph valuations. For this case, we show the existence of complete allocations that are both fair and efficient under the fairness notion of EFX$^+$ (a slight relaxation of EFX) and the efficiency notion of approximate maximum Nash welfare. Specifically, in the multigraph setting, we can decompose the fair division problem to one on $\binom{n}{2}$ smaller sub-problems, which can be solved with the established algorithms of [Lipton et al., 2004]. This algorithmic result on multigraphs nearly matches our demonstrated upperbounds on EFX$^+$ + MNW, where closing this small gap is a natural direction for future work. Additionally, as a direct extension on our problem, fair allocation on *hypergraphs* would be a natural next step. However, we highlight that even in the setting of 3-uniform hypergraphs the problem reduces to one of $\binom{n}{3}$ interconnected problem instances on 3 agents which is a highly non-trivial extension on the current existence results for small instances.

Part II

Fair Division in the Online Model

Chapter 4: Online Algorithms for the Santa Claus Problem

4.1 Introduction

Fair allocation of resources is one of the central themes of algorithmic fairness and game theory. In fact, the theory of fair division has its roots in mathematics going back to as early as 1948 [Steinhaus, 1948]. In the general setting, this problem comprises a set of items that must be divided among a set of agents in an egalitarian manner, where each agent has a (possibly non-uniform) valuation for each item. A natural objective to capture the goal of fair division is to maximize the minimum total value of items received by any agent. This gives rise to the famous "Santa Claus problem" that we describe below.

In the Santa Claus problem, originally described by Bansal and Sviridenko in 2006 [Bansal and Sviridenko, 2006] (although it was studied under different names or assumptions prior to this), the Santa Claus is said to have a set of m presents to be distributed equitably among n children. Each child $i \in [n]$ has some arbitrary non-negative value v_{ij} for present $j \in [m]$. Santa's goal is to distribute the presents in a way that makes the least satisfied child maximally satisfied. More formally, this means that the assignment seeks to maximize the minimum total value of the presents received by any child, where the total value of presents received by a child is the sum of her values for the presents that she received. The Santa Claus problem can be

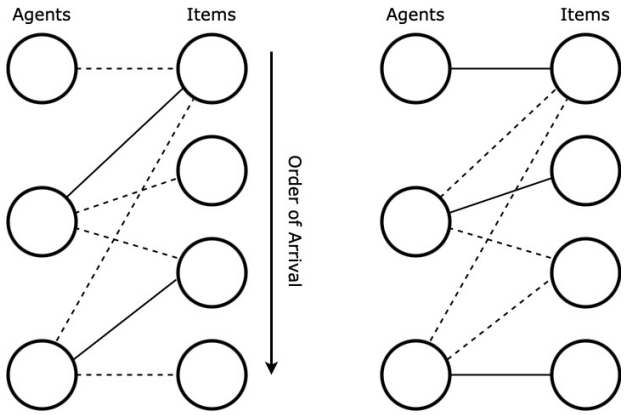


Figure 4.1: Example problem instance. The left panel shows a solution in the online input model, with the right panel depicting an offline optimal solution. Dashed lines indicate potential allocations, and solid lines indicate allocation decisions.

formalized as the following integer program:

$$\max \left\{ \min_{i \in [n]} \sum_{j=1}^{m} v_{ij} x_{ij} \mid \sum_{i=1}^{n} x_{ij} \leq 1 \ \forall j \in [m], \ x \in \{0,1\}^{mn} \right\}.$$

There is substantial literature going back more than 50 years that studies variants of this problem in the offline setting (see related work). In many practical situations, however, the set of items to be allocated is not known in advance. For example, in online advertising, ad-space providers will receive monetary bids from competing agents for the display of their advertisements in real-time for an available space on a webpage [Buchbinder et al., 2007, Blum et al., 2006, Hajiaghayi et al., 2007, Zhou et al., 2008]. The provider must then make irrevocable decisions as to which advertiser's bid to accept based only on knowledge of prior allocations and the current bids for the available space [Balseiro and Gur, 2019, Conitzer et al., 2021]. Beyond advertising, online allocation procedures have been useful in the study of donation distribution, wireless charging networks, organ donor matching, etc (see [Aleksandrov and Walsh, 2020] for a survey of these applications).

Motivated by these applications, we consider the *online* Santa Claus problem in this paper. In this setting, the items arrive in an online sequence and must be allocated to one of the agents immediately upon arrival. As in the offline problem, our goal is to maximize the minimum total value among all agents. To illustrate the problem, consider the simple example in Figure 4.1 where edges represent unit value of an agent for an item. At the time of the first arriving (leftmost) item, all agents can be matched to this item and therefore any agent is equally likely to receive it. However, as we continue forward with the input stream, we see in retrospect that the only nonzero max-min solution corresponds to the case where the first item is allocated to the first agent agent (as it is the only item for which they have a nonzero value). In the offline instance (depicted on the right), it is very easy to see this solution, but the incremental reveal of information requires both sophisticated algorithms and assumptions on the input instance that diminish the possibility of an optimal solution.

Following standard terminology for online algorithms (see, e.g., [Borodin and El-Yaniv, 2005]), we define the *competitive ratio* of an online algorithm as the minimum ratio between the value of the (maximization) objective in the algorithm's solution to that in the optimal (offline) solution in hindsight. We furthermore discuss the *additive regret* as the additive loss factor of our algorithm. More formally, we say our algorithm, ALG, has competitive ratio c and additive regret b if $\text{ALG} \geq c \cdot \text{OPT} - b$.

Prior work on the max-min objective in the online setting required various relaxations of the problem, such as allowing for some reordering in the allocation process [Epstein et al., 2011], restricting the number of agents [He and Jiang, 2005, Tan and Cao, 2006, Wu et al., 2014], or allowing migration of items after assignment [Chen and Qin, 2011]. This is because of two reasons. First, even in the offline setting, there remains a significant gap between the best upper and lower

bounds on the approximation ratio of the Santa Claus problem, and bridging this gap is a major open problem. Second, as we will soon see, there is a simple construction for the online problem that shows the competitive ratio cannot be better than n. To bypass these bottlenecks, our first assumption in this paper is that the items arrive in *random order*. This is a standard assumption that has been used to simplify many related online problems [Babaioff et al., 2007, Devanur and Hayes, 2009, Feldman et al., 2009a, Feldman et al., 2009b, Goel and Mehta, 2008, Karande et al., 2011, Kleinberg, 2005]. But, even with this assumption, we show that obtaining a competitive solution is impossible in general *for small problem instances*. This motivates our second assumption: that the objective value of the optimal solution is sufficiently large (with respect to the values of individual items). With these two assumptions, we give an algorithm that obtains a competitive ratio of $(1-\varepsilon)$ for any $\varepsilon > 0$. We note that using standard techniques, the assumption about the optimal objective being sufficiently large can be replaced by a corresponding additive regret in the competitive ratio.

We now formally define the two online input models that we consider in this paper: *adversarial* and *random order* input.

Definition 4.1.1 (Adversarial Input)**.** An adversary selects the value vector $v \in [0,1]^n$ of each arriving item for all the agents, as well as the order in which these vectors arrive.

Definition 4.1.2 (Random-Order Input)**.** An adversary selects the value vector $v \in [0,1]^n$ of each arriving item for all the agents, but these vectors are randomly permuted to determine their arrival order.

Note that in the literature, the independent and identically distributed (i.i.d.) input model is also often studied for related problems [Goel and Mehta, 2008, Karp et al., 1990, Mahdian and

Yan, 2011, Karande et al., 2011, Agrawal and Devanur, 2014, Kesselheim et al., 2014, Seiden, 2002, Molinaro and Ravi, 2014]. In this model, the adversary picks a distribution over inputs that is unknown to the algorithm and arriving items are sampled i.i.d. from this distribution. The random order model is *stronger* than the i.i.d. model in the sense that any algorithmic result for the random order model automatically extends to the i.i.d. setting as well. This includes the algorithmic results that we obtain in this paper for the random order arrival model.

4.1.1 Problem Definition

We here introduce the notation that we will use in the rest of the paper. Let $\mathbf{v}^1, ..., \mathbf{v}^m \in [0,1]^n$ denote the input sequence of items arriving in random order where v_i^t is the value of the t-th item to the i-th agent. We additionally denote by $\mathbf{x}^1, ..., \mathbf{x}^m \in [0,1]^n$ the fractional allocation of each item by the algorithm. We further let the corresponding allocations of a fixed (offline) optimal solution be denoted as $\overline{\mathbf{x}}^1, ..., \overline{\mathbf{x}}^m \in [0,1]^n$ and let OPT denote the max-min objective value of the optimal solution. For simplicity, we slightly abuse notation by letting $\mathbf{V}^t = (v_1^t x_1^t, \ldots, v_n^t x_n^t)$ and $\overline{\mathbf{V}}^t = (v_1^t \overline{x}_1^t, \ldots, v_n^t \overline{x}_n^t)$ for $t \in [m]$. Our algorithm thus seeks to maximize the coordinate-wise minimum of $\sum_{t=1}^m \mathbf{V}^t$.

4.1.2 Our Contributions

First, we give a simple construction to show that if the arrival order of the items is adversarial, then the best competitive ratio that can be achieved is only $1/n$. (In fact, we also match this competitive ratio using a simple algorithm in the supplementary material.)

Theorem 4.1.3 (Adversarial Input)**.** *In the adversarial setting, no algorithm can obtain a com-*

petitive ratio better than $1/n$.

This motivates us to consider the *random order model*, where an adversary again selects the set of items but they are then presented in random permutation order. For this setting, we give an algorithm that obtains a *fractional* assignment that is nearly optimal:

Theorem 4.1.4 (Random Order: Algorithm)**.** *For any* $\varepsilon > 0$*, there is an online fractional algorithm for the Santa Claus problem that has a competitive ratio of* $(1 - \varepsilon)$ *in the random order input model under the assumption that* $\text{OPT} \geq \Omega\left(\frac{\log n}{\varepsilon^2}\right)$.

We further show that, through randomized rounding, we can give an *integral* allocation that retains the near optimality of this fractional allocation.

Our complimentary result comes in the form of an impossibility result for algorithms in the random order input model. Specifically, we show that the lower bound on the value of OPT in the above theorem is *necessary*:

Theorem 4.1.5 (Random Order: Impossibility Result)**.** *For any* $\varepsilon \in (0, 1)$*, there is no online algorithm for the Santa Claus problem in the random order input model that has a competitive ratio of* $(1 - \varepsilon)$ *when* $\text{OPT} < C \cdot \frac{\ln n}{\varepsilon}$ *for some (absolute) constant* $C > 0$.

The reader will note that the lower bound on OPT is precise as a function of n, but there is a slight mismatch between the upper and lower bounds as a function of ε—bridging this gap is an interesting open question. We summarize our results for the *online* Santa Claus problem in Table 4.1.

Input Model	Algorithm	Competitive Ratio
Adversarial	RANDOM	$\left(\frac{1-\varepsilon}{n}\right)$ OPT $- O\left(\frac{n \log n}{\varepsilon^3}\right)$
Random Order	GREEDYWR	$(1-\varepsilon)$OPT $- O\left(\frac{\log n}{\varepsilon}\right)$

Table 4.1: Results for the Santa Claus problem in the online input model.

4.1.3 Related Work

The general case of the Santa Claus problem was initially explored (under a different name) in the field of algorithmic game theory for the fair allocation of goods [Lipton et al., 2004]. By studying the assignment LP of [Lenstra et al., 1990] for the dual "makespan" problem, Bezakova and Dani [Bezáková and Dani, 2005] derived an additive approximation of $\max_{ij} v_{ij}$, i.e., the objective in the algorithm's solution is at least OPT $- \max_{ij} v_{ij}$ where OPT is the objective value of the optimal solution. They also extended the hardness result on the dual makespan minimization problem [Lenstra et al., 1990] to demonstrate that the Santa Claus problem is NP-hard and cannot be approximated to a factor better than 2. Later, Bansal and Sviridenko [Bansal and Sviridenko, 2006] demonstrated that the integrality gap of the configuration LP for this problem is $\Omega(\sqrt{n})$, while Asadpour and Saberi [Asadpour and Saberi, 2010] complimented this result with a $O(\sqrt{m}\log^3 m)$ upper bound for the same LP relaxation. To date, the best algorithmic result for the Santa Claus is an $\tilde{O}(n^{\varepsilon})$-approximate algorithm, where $\varepsilon = \Omega(\log\log n/\log n)$, in quasi-polynomial time obtained by Chakrabarty, Chuzhoy, and Khanna [Chakrabarty et al., 2009].

For the special case of *restricted assignment*, i.e. $v_{ij} \in \{0, v_j\}$, Bansal and Sviridenko [Bansal and Sviridenko, 2006] provided an $\Omega\left(\log\log\log m/\log\log m\right)$-approximate algorithm that relies on rounding a configuration LP. Later, Feige gave a non-constructive proof that this LP relaxation was within a constant factor of OPT [Feige, 2008]. Asadpour, Feige, and Saberi [Asad-

pour et al., 2012] made this constructive, obtaining a $1/4$-approximation via a rounding algorithm based on local search, but the algorithm is not known to converge in polynomial time. Further work has since improved this constant factor [Jansen and Rohwedder, 2020, Cheng and Mao, 2018, Davies et al., 2020, Annamalai et al., 2017], improved upon the running time [Cheng and Mao, 2018], and extended the setting beyond additive valuations [Bamas et al., 2025].

Online Assignment. The study of online assignment is expansive, but much of the classical work is for adversarial arrival order. Even in the random order setting, a broad range of problems have been considered in recent years including the secretary problem [Babaioff et al., 2007, Kleinberg, 2005], AdWords [Devanur and Hayes, 2009, Feldman et al., 2009a, Goel and Mehta, 2008], online matching [Feldman et al., 2009b, Karande et al., 2011], online packing [Gupta and Molinaro, 2014, Feldman et al., 2010, Kesselheim et al., 2014], online scheduling [Molinaro, 2017, Liu et al., 2021], etc. One example of max-min online assignment in the random order setting is the work of Gollapudi and Panigrahi [Gollapudi and Panigrahi, 2014] who considered revenue maximization with fairness objectives. Another related work is that of Molinaro [Molinaro, 2017] for the dual min-max objective, who builds on prior work leveraging the experts framework from online learning [Gupta and Molinaro, 2014] to give algorithms that simultaneously perform well in the adversarial and random-order settings. A third line of work relevant to our paper is that of online packing problems in the random arrival order (e.g., [Feldman et al., 2010, Agrawal and Devanur, 2014, Kesselheim et al., 2014, Seiden, 2002, Molinaro and Ravi, 2014]. In particular, the results of Agrawal *et al.* [Agrawal and Devanur, 2014] have a similar flavor to ours: they obtain $(1 - \varepsilon)$-competitiveness assuming a large enough optimal value for the online packing problem in the random order setting.

4.2 Online Algorithm for the Santa Claus Problem in the Random Order Model

In this section, we present the approximately greedy algorithm, Algorithm 6, and analyze its competitive ratio in the random order model. Building on the work of Molinaro for online scheduling [Molinaro, 2017], we use a greedy algorithm for a smoothed version of our objective function and a restart procedure during the online allocation process to reduce the impact of correlations that arise in this input model.

A natural strategy for our problem is to allocate the arriving item to the least satisfied agent. However, one can show this strategy has too high an additive regret [Gupta and Molinaro, 2014, Molinaro, 2017] as the change in our solution value can vary quickly from one iteration to the next. Instead, we use a *smoothed* version of the greedy algorithm. The algorithm is designed as follows: we first define ϕ_ε to be a re-scaled variant of the LogSumExp function that serves as a smoothed minimum. For the first half of the input stream, we select an allocation for each arriving item that maximizes the increase in our smoothed objective function. This stage can be thought of as approximately greedy with respect to the *gradient* of ϕ_ε. After $\frac{m}{2}$ items have been allocated, we "restart" the allocation by maximizing the increase in our objective with respect to the $t > \frac{m}{2}$ allocations only. This restart procedure is essential for reducing the correlations that arise in sampling without replacement in the random order model since at each iteration of the allocation procedure, our decision depends on at most $\frac{m}{2} - 1$ items. The pseudocode of this procedure is presented in Algorithm 6.

Algorithm 6 SMOOTH GREEDY WITH RESTART

Require: $0 < \varepsilon < 1$, input stream $\mathcal{M}$ of m items
Ensure: $(1-\varepsilon)$-competitive MAXMIN allocation
1: Define $\phi_\varepsilon(u) = -\frac{1}{\varepsilon}\ln\left(\sum_i e^{-\varepsilon u_i}\right)$
2: **for** $t = 1$ to $\frac{m}{2}$ **do**
3: Select $\mathbf{x}^t \in \Delta^n$ to maximize $\phi_\varepsilon\left(\sum_{\tau=1}^{t}\mathbf{V}^\tau\right)$
4: **end for**
5: **for** $t = \frac{m}{2}+1$ to m **do**
6: Select $\mathbf{x}^t \in \Delta^n$ to maximize $\phi_\varepsilon\left(\sum_{\tau=\frac{m}{2}+1}^{t}\mathbf{V}^\tau\right)$
7: **end for**

4.2.1 Algorithm Analysis

In the analysis of the competitive ratio of Algorithm 6, we will leverage several key facts about the smoothed minimum function ϕ_ε. This smoothness implies that the gradient nicely captures incremental increase in the objective function, thus allocating with respect to this produces an essentially greedy process, allowing us to follow the analysis of [Agrawal and Devanur, 2014, Devanur and Hayes, 2009, Molinaro, 2017] to get the desired guarantees for the random-order model.We now state our main result in Theorem 4.2.1.

Theorem 4.2.1. *For any $\varepsilon > 0$, Algorithm 6 guarantees in the random-order input model that the expected value of the allocation assigned to any agent is at least*

$$(1-\varepsilon)\cdot \text{OPT} - O\left(\frac{\log n}{\varepsilon}\right).$$

Note that this theorem immediately implies the following corollary since the additive regret term can be absorbed in the multiplicative error for sufficiently large OPT:

Corollary 4.2.2. *For any $\varepsilon > 0$, Algorithm 6 has a competitive ratio of $(1-\varepsilon)$ for* $\text{OPT} \geq \Omega(\frac{\log n}{\varepsilon^2})$.

In order to prove this theorem, we first show some properties of the smoothed minimum function ϕ_ε utilized by Algorithm 6. We will then prove two technical lemmas that will help establish the theorem. Specifically, Lemma 4.2.3 effectively defines the additive error with respect to our true objective, the agent-wise minimum, and stability of the smooth function following each allocation decision. As was shown in prior work, allocating with respect to the order statistics or even the L_p^p norm produces either too high of a regret factor, or instability in the derivative value under small perturbations to the input value. As such, the smoothing and the following properties are critical to maintaining our regret and competitive ratio bounds.

Lemma 4.2.3. *For all $u \in \mathbb{R}^n$, $v \in [0,1]^n$, and $\varepsilon > 0$, the function $\phi_\varepsilon(x) = -\frac{1}{\varepsilon}\ln\left(\sum_{i=1}^n e^{-\varepsilon x_i}\right)$ satisfies the following:*

(a) $\min_i\{u_i\} - \frac{\ln n}{\varepsilon} \leq \phi_\varepsilon(u) < \min_i\{u_i\}$

(b) $\nabla\phi_\varepsilon(u+v) \in e^{\pm\varepsilon} \cdot \nabla\phi_\varepsilon(u)$

Furthermore, if $u_i \geq v_i$ for each $i \in [n]$, we have

(c) $\phi_\varepsilon(u-v) \leq \phi_\varepsilon(u) - \phi_\varepsilon(v)$

Now utilizing these properties we can prove the following important bound on the inner product of the smoothed minimum's gradient. This will be used throughout our analysis to bound the incremental change in the objective MAXMIN value after each allocation decision, and the summation of these changes can be seen as the accumulated "reward" at any given stage of the input stream. The proof is by direct integration of stability property (b).

Lemma 4.2.4. *For $u \in \mathbb{R}^n$ and $v, v' \in [0,1]^n$, if $\phi_\varepsilon(u+v) \geq \phi_\varepsilon(u+v')$ then $\langle\nabla\phi_\varepsilon(u), v\rangle \geq e^{-2\varepsilon}\langle\nabla\phi_\varepsilon(u), v'\rangle$.*

We follow in the intuition of Agrawal and Devanur [Agrawal and Devanur, 2014] to prove Theorem 4.2.1 by bounding the incremental increase in our "reward" both before and after the restart at $t = \frac{m}{2}$. By implementing this restart in the allocation procedure, we segment the stream into two portions that have identical probabilistic guarantees and reduce the correlation between input elements to allow for an optimal competitive ratio and low additive regret.

Lastly, since the previous algorithm produces an online fractional solution for the Santa Claus problem, we further show that using simple randomized rounding, we can convert this into an integer solution.

Theorem 4.2.5. *Fix any $\varepsilon > 0$. Given a $(1-\varepsilon)$competitive online fractional algorithm for the Santa Claus problem, there is an online (integral) algorithm whose competitive ratio is $(1-O(\varepsilon))$, provided* OPT $\geq \Omega\left(\frac{\log n}{\varepsilon^2}\right)$.

4.3 Random Order Lower Bound

We here present an impossibility result on the Santa Claus problem under the random order input models.

Theorem 4.3.1. *For any $\gamma \in (0,1)$, if a randomized algorithm* ALG*for the online Santa Claus problem with random order input satisfies*

$$\text{ALG} \geq (1-\gamma) \cdot \text{OPT},$$

then OPT $\geq \frac{C \ln n}{\gamma}$ *for some (absolute) constant $C > 0$.*

We use the following construction, which extends on the proof of Theorem 4.1.3. There

are n agents, of which $n-1$ are *private* agents and the remaining one is a *public* agent. Every private agent has k distinct private items for which their valuation is 1 each, and the valuation for every other agent is 0. In addition, there are k public items, each of which has a valuation of 1 for every agent. The optimal solution is to assign the private items to the corresponding private agents, and the public items to the public agent. Thus, OPT $= k$.

Now, when the items are presented in uniform random order, consider the first ε fraction of presented items—call this the ε-prefix. Our main claim is that with constant probability, the following statements both hold:

(a) there are about ε fraction of public items in this ε-prefix

(b) there is at least one type of private item that is missing from this ε-prefix.

To verify property (a), we leverage the following concentration inequality from Devanur and Hayes [Devanur and Hayes, 2009]:

Lemma 4.3.2 (Lemma 3 in [Devanur and Hayes, 2009])**.** *Let $Y = (Y_1, \ldots, Y_m)$ be a vector of real numbers, and let $\varepsilon \in (0,1)$. Let S be a random subset of $[m]$ of size εm, and set $Y_S := \sum_{j \in S} Y_j$. Then, for every $\delta \in (0,1)$,*

$$\Pr\left[\,|Y_S - \mathbb{E}\,[\,Y_S\,]| \geq \frac{2}{3}\|Y\|_\infty \ln\left(\frac{2}{\delta}\right) + \|Y\|_2\sqrt{2\varepsilon \ln\left(\frac{2}{\delta}\right)}\,\right] \leq \delta.$$

Property (a) concerning the fraction of public items in the ε-prefix follows almost immediately from the above lemma:

Lemma 4.3.3. *The probability that there are fewer than $\frac{5\varepsilon}{12}$ fraction of public items in the ε-prefix is at most $2\exp\left(\frac{-\varepsilon k}{8}\right)$.*

We then proceed to verify property (b) in order to demonstrate the non-existence of at least one type of private item in the ε-prefix.

Lemma 4.3.4. *The probability that all the $n-1$ types of private items appear in the ε-prefix is at most* $\exp\left(-(n-1)\cdot 4^{\frac{-\varepsilon k}{1-\varepsilon}}\right)$.

Combining the two results, we can then tune k to show that both events hold (independently) with probability at most $\frac{1}{e}$. This further implies that with probability $1-\frac{2}{e}$, there will be a private item in the ε-prefix which the algorithm cannot identify as being a private item. As a consequence, we show that $\Omega(\varepsilon)$ public items are not allocated to the public agent, which means that the total valuation for this agent must be at most $(1-\Omega(\varepsilon))k$. Combining this with the fact that OPT $= k$, we complete the proof.

4.4 Proofs

4.4.1 Adversarial Lower Bounds

Proof of 4.1.3. We consider the following instance: of the n agents, $n-1$ are *private* agents each having k private items that have valuation 1 for the corresponding private agent and 0 for every other agent. The n-th agent is a *public* agent and there are k public items that have a valuation of 1 for all the n agents.

The optimal solution assigns the private items to the corresponding private agents and the public items to the public agent. Thus, OPT $= k$. In the online instance, the adversary chooses to present all the public items before the private items. Since all the agents look identical before the arrival of the first private item, the public agent gets no more than k/n items in expectation for

any algorithm (this adversarial input is comparable to our toy example discussed in Figure 4.1). Since none of the remaining private items can be allocated to the one public agent, their bundle value can no longer increase beyond k/n. The theorem follows. □

Theorem 4.4.1. *In the adversarial setting, for any $\varepsilon \in (0, 1)$, there is an algorithm for the online Santa Claus problem that has a competitive ratio of $(1 - \varepsilon) \cdot \frac{1}{n}$ for $\text{OPT} > C \cdot \frac{n \ln n}{\varepsilon^2}$ for a large enough constant C.*

Proof. The algorithm is to simply assign every item uniformly at random among all the n agents. Note that for any fixed agent, its expected value is at least $\frac{1}{n} \cdot \text{OPT}$. By Chernoff bounds, the probability that its total value is less than $(1 - \frac{\varepsilon}{2})\frac{1}{n} \cdot \text{OPT}$ is given by $\exp(-\frac{1}{2} \cdot \frac{\varepsilon^2}{4} \cdot \frac{1}{n} \cdot \text{OPT}) < \frac{\varepsilon}{2n}$ for $\text{OPT} > C \cdot \frac{n \ln n}{\varepsilon^2}$ for a sufficiently large constant C. By union bound over all the n agents, the probability that *any* agent's overall value is less than $(1 - \frac{\varepsilon}{2})\frac{1}{n} \cdot \text{OPT}$ is at most $\frac{\varepsilon}{2}$. Thus, the expected competitive ratio is at least $(1 - \frac{\varepsilon}{2})(1 - \frac{\varepsilon}{2}) \cdot \frac{1}{n} > (1 - \varepsilon) \cdot \frac{1}{n}$. □

4.4.2 Algorithm Analysis

Proof of Lemma 4.2.3. Property (a) is an established property of the LogSumExp function [Blanchard et al., 2021, Nielsen and Sun, 2016] but we present the proof here for completeness. We start by exponentiating the input, summing over all elements and applying the logarithm to

the resultant bounds.

$$\begin{aligned}
&\exp \max_i\{-\varepsilon u_i\} \leq \sum_{i=1}^{n} \exp -\varepsilon u_i < n \exp \max_i\{-\varepsilon u_i\} \\
&\overset{(i)}{\iff} \max_i\{-\varepsilon u_i\} \leq \ln \sum_{i=1}^{n} \exp -\varepsilon u_i < \max_i\{-\varepsilon u_i\} + \ln n \\
&\overset{(ii)}{\iff} -\max_i\{-\varepsilon u_i\} > -\ln \sum_{i=1}^{n} \exp -\varepsilon u_i \geq -\max_i\{-\varepsilon u_i\} - \ln n \\
&\overset{(iii)}{\iff} \min_i\{\varepsilon u_i\} > -\ln \sum_{i=1}^{n} \exp -\varepsilon u_i \geq \min_i\{\varepsilon u_i\} - \ln n
\end{aligned}$$

Where (i) is the result of taking the logarithm, (ii) is a negation on the inequalities and (iii) is by property of the maximum. Now, since $\varepsilon > 0$, the result follows from simple algebraic manipulation.

$$\begin{aligned}
&\min_i\{\varepsilon u_i\} > -\ln \sum_{i=1}^{n} \exp -\varepsilon u_i \geq \min_i\{\varepsilon u_i\} - \ln n \\
&\overset{(iv)}{\iff} \varepsilon \min_i\{u_i\} > -\ln \sum_{i=1}^{n} \exp -\varepsilon u_i \geq \varepsilon \min_i\{u_i\} - \ln n
\end{aligned}$$

$$\begin{aligned}
&\overset{(v)}{\iff} \min_i\{u_i\} > \frac{-1}{\varepsilon} \ln \sum_{i=1}^{n} \exp -\varepsilon u_i \geq \min_i\{u_i\} - \frac{\ln n}{\varepsilon} \\
&\overset{(vi)}{\iff} \min_i\{u_i\} > \phi_\varepsilon(u) \geq \min_i\{u_i\} - \frac{\ln n}{\varepsilon}
\end{aligned}$$

where (iv) follows from positive scalar multiplication within a minimum, (v) by dividing through by ε and (vi) is merely the definition of our smoothing function ϕ_ε. This verifies the desired property.

To prove (b), we first calculate the partial derivative of the smoothed minimum function

$$\frac{\partial}{\partial x_i}\phi_\varepsilon(u) = \frac{e^{-\varepsilon u_i}}{\sum_{j=1}^n e^{-\varepsilon u_j}}$$

and now, using $u_i \geq 0$ and $v_i \in [0,1]$ we derive

$$\begin{aligned}
&\frac{e^{-\varepsilon}e^{-\varepsilon u_i}}{\sum_{j=1}^n e^{-\varepsilon u_j}} < \frac{e^{-\varepsilon(u_i+v_i)}}{\sum_{j=1}^n e^{-\varepsilon(u_j+v_j)}} < \frac{e^{-\varepsilon u_i}}{e^{-\varepsilon}\sum_{j=1}^n e^{-\varepsilon u_j}} \\
&\iff e^{-\varepsilon}\frac{\partial}{\partial x_i}\phi_\varepsilon(u) < \frac{\partial}{\partial x_i}\phi_\varepsilon(u+v) < e^{\varepsilon}\frac{\partial}{\partial x_i}\phi_\varepsilon(u)
\end{aligned}$$

Therefore, we have property (b). Lastly, to prove (c) we first invoke the definition ϕ_ε:

$$\frac{-1}{\varepsilon}\ln\left(\sum_{i=1}^n e^{-\varepsilon(x_i-y_i)}\right) \leq \frac{-1}{\varepsilon}\left(\ln\left(\sum_{i=1}^n e^{-\varepsilon x_i}\right) - \ln\left(\sum_{i=1}^n e^{-\varepsilon y_i}\right)\right).$$

This statement is equivalent to

$$\ln\left(\sum_{i=1}^n e^{-\varepsilon(x_i-y_i)}\right) \geq \ln\left(\sum_{i=1}^n e^{-\varepsilon x_i}\right) - \ln\left(\sum_{i=1}^n e^{-\varepsilon y_i}\right)$$

which, by exponentiating both sides, yields

$$\begin{aligned}
&\sum_{i=1}^n e^{-\varepsilon(x_i-y_i)} \geq \left(\sum_{i=1}^n e^{-\varepsilon x_i}\right)\cdot\left(\sum_{i=1}^n e^{-\varepsilon y_i}\right)^{-1} \\
&\iff \sum_{i=1}^n e^{-\varepsilon(x_i-y_i)}\cdot\left(\sum_{i=1}^n e^{-\varepsilon y_i}\right) \geq \left(\sum_{i=1}^n e^{-\varepsilon x_i}\right)
\end{aligned}$$

and expansion of the left-hand side verifies the claim. □

Proof of Lemma 4.2.4. By direct integration and the stability property (b), we see

$$\begin{aligned}\phi_\varepsilon(u+v) &= \phi_\varepsilon(u) + \int_0^1 \langle \nabla\phi_\varepsilon(u+\alpha v), v\rangle d\alpha \\ &\in \phi_\varepsilon(u) + e^{\pm\varepsilon}\langle \nabla\phi_\varepsilon(u), v\rangle\end{aligned}$$

Therefore, we can further show

$$\begin{aligned}\langle \nabla\phi_\varepsilon(u), v\rangle &\geq e^{-\varepsilon}[\phi_\varepsilon(u+v) - \phi(u)] \\ &\geq e^{-\varepsilon}[\phi_\varepsilon(u+v') - \phi(u)] && \text{(Input assumption)} \\ &\geq e^{-2\varepsilon}\langle \nabla\phi_\varepsilon(u), v'\rangle\end{aligned}$$

where the first and last inequality is a direct result of the integration above. □

Proof of Theorem 4.2.1. We note again that in the approximately greedy procedure of Algorithm 6, we are essentially seleting items greedily according to the *gradient* of ϕ_ε to maximize the incremental changes. Due to the restart at $m/2$, we define

$$\nabla^t = \nabla\phi_\varepsilon\left(\sum_{\tau=1}^{t-1}\mathbf{V}^\tau\right) \text{ for } t \leq \frac{m}{2} \text{ and } \nabla^t = \nabla\phi_\varepsilon\left(\sum_{\tau=m/2+1}^{m}\mathbf{V}^\tau\right) \text{ for } t > \frac{m}{2}$$

We now proceed by deriving a bound on $\min_i\{\sum_{\tau=1}^m \mathbf{V}_i^\tau\}$ in terms of our smoothed-approximation function ϕ_ε, thus bounding the accumulated error by the algorithm when estimating our true ob-

jective. By the concavity of ϕ_ε and Lemma 4.2.4, we have that

$$\begin{aligned}
\phi_\varepsilon\left(\sum_{\tau=1}^{t}\mathbf{V}^\tau\right)-\phi_\varepsilon\left(\sum_{\tau=1}^{t-1}\mathbf{V}^\tau\right) &\geq \left\langle \nabla\phi_\varepsilon\left(\sum_{\tau=1}^{t}\mathbf{V}^\tau\right),\mathbf{V}^t\right\rangle && \text{(Concavity)}\\
&\geq e^{-\varepsilon}\left\langle \nabla\phi_\varepsilon\left(\sum_{\tau=1}^{t-1}\mathbf{V}^\tau\right),\mathbf{V}^t\right\rangle && \text{(Lemma 4.2.4)}\\
&= e^{-\varepsilon}\langle\nabla^t,\mathbf{V}^t\rangle.
\end{aligned}$$

Without loss of generality we proceed by considering the first half of the input sequence ($t \leq \frac{m}{2}$). By summing this inequality over the input prior to the restart (from $t=1$ to $\frac{m}{2}$), we have

$$\begin{aligned}
\phi_\varepsilon\left(\sum_{\tau=1}^{m/2}\mathbf{V}^\tau\right)-\phi_\varepsilon(0) &= \phi_\varepsilon\left(\sum_{\tau=1}^{m/2}\mathbf{V}^\tau\right)+\frac{\ln n}{\varepsilon} && (4.1)\\
&\geq e^{-\varepsilon}\sum_{t=1}^{m/2}\langle\nabla^t,\mathbf{V}^t\rangle && (4.2)
\end{aligned}$$

Now, taking into account the allocation before and after the restart at $\frac{m}{2}$ and invoking Eq. 4.2 for each half of the input stream with the concavity of ϕ_ε, we obtain

$$\begin{aligned}
\sum_{t=1}^{m}\langle\nabla^t,\mathbf{V}^t\rangle &\leq e^{\varepsilon}\left(\phi_\varepsilon\left(\sum_{\tau=1}^{m/2}\mathbf{V}^\tau\right)+\phi_\varepsilon\left(\sum_{\tau=m/2+1}^{m}\mathbf{V}^\tau\right)+\frac{2\ln n}{\varepsilon}\right) && \text{(Ineq. 4.2)}\\
&\leq e^{\varepsilon}\left(\phi_\varepsilon\left(\sum_{\tau=1}^{m}\mathbf{V}^\tau\right)+\frac{2\ln n}{\varepsilon}\right) && \text{(Lemma 4.2.3c)}\\
&\leq e^{\varepsilon}\left(\min_i\left\{\sum_{\tau=1}^{m}\mathbf{V}_i^\tau\right\}+\frac{2\ln n}{\varepsilon}\right) && \text{(Lemma 4.2.3a)}
\end{aligned}$$

Lastly, by rearranging terms we can bound the element-wise minimum as

$$\min_i \left\{ \sum_{\tau=1}^{m} \mathbf{V}_i^{\tau} \right\} \geq e^{-\varepsilon} \left(\sum_{t=1}^{m} \langle \nabla^t, \mathbf{V}^t \rangle \right) - \frac{2 \ln n}{\varepsilon}. \tag{4.3}$$

Thus, the output of our algorithm will approximate the *actual* objective function within a multiplicative factor of $e^{-\varepsilon} \approx 1 - \varepsilon$ and an additive regret on the order of $\ln n / \varepsilon$.

We now proceed to bound the gap between our algorithmic solution to that of the optimal *offline* solution which, combined with the above error, will give the final result. As such, the final step of our analysis will be to take the expectation of this inequality to get the final bounds in Theorem 4.2.1. Due to the restart and random order input model, the expected increase in maximal value over the first and second half is equivalent,

$$\mathbb{E}\left[\sum_{t=1}^{m/2} \langle \nabla^t, \mathbf{V}^t \rangle \right] = \mathbb{E}\left[\sum_{t=\frac{m}{2}+1}^{m} \langle \nabla^t, \mathbf{V}^t \rangle \right]$$

so without loss of generality we need only bound the first half's value and apply this bound to both portions. The benefit of this restart will yield a tolerable error as compared to the optimal offline solution in each half of the allocation procedure, rather than a continuously accumulating divergence between the two solutions: each arriving job's allocation is only dependent upon (at most) $\frac{n}{2} - 1$ other decisions. We leverage this randomness to obtain the final competitive guarantees [Molinaro and Ravi, 2014].

Now, by the nature of greedy selection to maximize our objective function, we must have that in each iteration our algorithm's selection produces an increase in value that is at least as

good as that of the optimal offline solution: $\phi_\epsilon\left(\sum_{\tau=1}^{t}\mathbf{V}^\tau\right) \geq \phi_\epsilon\left(\sum_{\tau=1}^{t-1}\mathbf{V}^\tau + \overline{\mathbf{V}}^t\right)$ for each t[1]. Combining this with Lemma 4.2.4, we must have

$$\langle \nabla^t, \mathbf{V}^t \rangle \geq e^{-2\varepsilon}\langle \nabla^t, \overline{\mathbf{V}}^t \rangle$$

Furthermore, since $\nabla\phi_\varepsilon \in \ell_1^+$ [Gao and Pavel, 2017] and ∇^t is independent of $\mathbf{V}^t$, we can prove the following purely probabilistic result that will later be used to bound the above summations to derive the final result.

Lemma 4.4.2. *Consider a set of vectors $\{y^1, ..., y^m\} \in [0,1]^n$, let $\{\mathbf{Y}^i\}_{i=1}^k$ be sampled without replacement. Let $\mathbf{Z} \in \ell_1^+$ be a random vector that depends only on $\{\mathbf{Y}^i\}_{i=1}^{k-1}$. Then for all $\varepsilon > 0$,*

$$\mathbb{E}\left[\,\langle \mathbf{Y}^k, \mathbf{Z}\rangle\,\right] \geq e^{-\varepsilon}\min_i\left\{\mathbb{E}\left[\,\mathbf{Y}_i^k\,\right]\right\} - \frac{\ln n}{\varepsilon\,(m-k+1)}.$$

Using this lower bounding result, we can now bound the sum of rewards as

$$\mathbb{E}\left[\,\langle \nabla^t, \overline{\mathbf{V}}^t\rangle\,\right] \geq e^{-\varepsilon}\min_i\left\{\mathbb{E}\left[\,\overline{\mathbf{V}}_i^t\,\right]\right\} - \frac{\ln n}{\varepsilon(m-t+1)}. \tag{4.4}$$

Adding this inequality over all $t \leq \frac{m}{2}$ in combination with $\min_i\{\mathbb{E}\left[\,\overline{\mathbf{V}}_i^t\,\right]\} = \frac{\text{OPT}}{m}$ we conclude

[1]The optimal offline algorithm may allocate in a manner that does not maximize the objective function's increase at iteration t in anticipation of better allocation options later in the input sequence.

that

$$
\begin{aligned}
e^{2\varepsilon} \cdot \mathbb{E}\left[\sum_{t\le\frac{m}{2}}\langle\nabla^t, \mathbf{V}^t\rangle\right] &\ge \mathbb{E}\left[\sum_{t\le\frac{m}{2}}\langle\nabla^t, \overline{\mathbf{V}}^t\rangle\right] && \text{(Lemma 4.2.3)}\\
&\ge e^{-\varepsilon}\sum_{t\le\frac{m}{2}}\min_i\left\{\mathbb{E}\left[\overline{\mathbf{V}}_i^t\right]\right\} - \sum_{t\le\frac{m}{2}}\frac{\ln n}{\varepsilon(m-t+1)} && \text{(Ineq. 4.4)}\\
&\ge e^{-\varepsilon}\left(\frac{\text{OPT}}{2}\right) - \frac{\ln n}{\varepsilon}
\end{aligned}
$$

where the final inequality is by expansion of the harmonic sum. Due to the restart at $t = \frac{m}{2}$, we can extend the sum to $t = m$ by simply doubling the above RHS. Finally, invoking inequality (4.3), we see that

$$
\mathbb{E}\left[\min_i\sum_{\tau=1}^{m}\mathbf{V}^\tau\right] \ge e^{-4\varepsilon}\text{OPT} - O\left(\frac{\log n}{\varepsilon}\right) \ge (1 - O(\varepsilon))\,\text{OPT} - O\left(\frac{\log n}{\varepsilon}\right).
$$

by the Taylor approximation $e^{-x} \ge 1 - x$. □

We lastly prove the key lemma.

Proof of Lemma 4.4.2. Let $\mu = \frac{1}{m}\sum_t y^t$ be the average of the set of vectors and, to simplify notation, we additionally break apart the expectation over selection of the k sets and let $\mathbb{E}\left[\langle\mathbf{Y}^k, \mathbf{Z}\rangle\right] = \mathbb{E}\left[\mathbb{E}_{k-1}\left[\langle\mathbf{Y}^k, \mathbf{Z}\rangle\right]\right]$ where $\mathbb{E}_{k-1}$ denote the expectation conditioned on $\mathbf{Y}^1, ..., \mathbf{Y}^{k-1}$. Note that since $\mathbf{Z}$ is a unit-vector in ℓ_1^+, an innerproduct of this vector with $\mathbf{Y}^k$ is simply a weighted sum of the latter's elements. Thus, we have

$$
\mathbb{E}_{k-1}\left[\langle\mathbf{Y}^k, Z\rangle\right] \ge \min_i\left\{\mathbb{E}\left[\mathbf{Y}^k|\mathbf{Y}^1, ..., \mathbf{Y}^{k-1}\right]\right\}. \tag{4.5}
$$

Additionally, by nature of the sampling set and the procedure of sampling without replacement, we have the conditional expectation

$$\mathbb{E}\left[\,\mathbf{Y}^k|\mathbf{Y}^1,...,\mathbf{Y}^{k-1}\,\right] = \frac{m\mu - (\mathbf{Y}^1 + ... + \mathbf{Y}^{k-1})}{m-(k-1)}$$

and further note that $m\mu - (\mathbf{Y}^1 + ... + \mathbf{Y}^{k-1})$ has the *same* distribution as $\mathbf{Y}^1 + ... + \mathbf{Y}^{m-(k-1)}$. This concretely gives us the simplifying equivalences

$$\mathbb{E}\left[\,\mathbf{Y}^k|\mathbf{Y}^1,...,\mathbf{Y}^{k-1}\,\right] = \frac{m\mu - (\mathbf{Y}^1 + ... + \mathbf{Y}^{k-1})}{m-(k-1)} = \frac{\sum_{t=1}^{m-(k-1)}\mathbf{Y}^t}{m-(k-1)}.$$

We now return to the inequality bound of (4.5) and, using the above equivalences, obtain

$$\begin{aligned}
\mathbb{E}\left[\min_i\left\{\sum_{t=1}^{m-(k-1)}\mathbf{Y}_i^t\right\}\right] &\geq \mathbb{E}\left[\,\phi_\varepsilon\left(\Sigma_{t=1}^{m-(k-1)}\mathbf{Y}^t\right)\right] && \text{(Lemma 4.2.3)}\\
&= \mathbb{E}\left[\frac{-1}{\varepsilon}\ln\left(\sum_i \exp\left(-\varepsilon\Sigma_{t=1}^{m-(k-1)}\mathbf{Y}_i^t\right)\right)\right]\\
&\geq \frac{-1}{\varepsilon}\ln\left(\sum_i \exp\left(-\varepsilon\mathbb{E}\left[\,\Sigma_{t=1}^{m-(k-1)}\mathbf{Y}_i^t\,\right]\right)\right) && \text{(Jensen's Ineq.)}\\
&\geq e^{-\varepsilon}\min_i\left\{\mathbb{E}\left[\,\Sigma_{t=1}^{m-(k-1)}\mathbf{Y}_i^t\,\right]\right\} - \frac{\ln n}{\varepsilon} && \text{(Lemma 4.2.3)}\\
&\geq e^{-\varepsilon}\min_i\left\{(m-(k-1))\mathbb{E}\left[\,\mathbf{Y}_i^t\,\right]\right\} - \frac{\ln n}{\varepsilon}
\end{aligned}$$

Finally combining the above results, we obtain

$$\begin{aligned}
\frac{e^{-\varepsilon}\min_i\left\{(m-(k-1))\mathbb{E}\left[\,\mathbf{Y}^t\,\right]\right\} - \frac{\ln n}{\varepsilon}}{m-(k-1)} &\leq \mathbb{E}\left[\,\min_i\{\mathbb{E}\left[\,\mathbf{Y}_i^k|\mathbf{Y}^1,...,\mathbf{Y}^{k-1}\,\right]\}\,\right]\\
&\leq \mathbb{E}\left[\,\langle\mathbf{Y}^t,\mathbf{Z}\rangle\,\right]
\end{aligned}$$

Thus, after rearranging terms, this completes the lemma. □

Proof of Theorem 4.2.5. The algorithm is simply randomized rounding. If an item j is allocated with fraction x_{ij} to agent i such that $\sum_{i=1}^{n} x_{ij} \leq 1$ by the fractional solution, then we assign item i to agent j with probability x_{ij}. Note that since $v_{ij} \in [0,1]$, the fractional value derived by an agent i from an item j, given by $v_{ij}x_{ij}$ is also in $[0,1]$. Thus, by Chernoff bounds, the probability that the total value of agent i in the rounded assignment is less than $(1-\varepsilon)\sum_{j=1}^{m} v_{ij}x_{ij}$ is at most

$$\exp\left(-\frac{\varepsilon^2}{3}\cdot\sum_{j=1}^{m} v_{ij}x_{ij}\right) \leq \exp\left(-\frac{\varepsilon^2}{3}\cdot(1-\varepsilon)\cdot \text{OPT}\right) \leq \frac{1}{n^2}, \text{ for OPT} \geq \Omega\left(\frac{\log n}{\varepsilon^2}\right).$$

Thus, with probability at least $1-\frac{1}{n^2}$, we have

$$(1-\varepsilon)\sum_{j=1}^{m} v_{ij}x_{ij} \geq (1-\varepsilon)^2\cdot\text{OPT} > (1-2\varepsilon)\cdot\text{OPT}.$$

It follows that the probability that the total value of agent i in the rounded assignment is less than $(1-2\varepsilon)\cdot\text{OPT}$ is at most $\frac{1}{n^2}$. Using the union bound over the n agents, we get that the probability that the total value of *any* agent in the rounded assignment is less than $(1-2\varepsilon)\cdot\text{OPT}$ is at most $\frac{1}{n}$. Thus, the expected value of the objective is at least $(1-\frac{1}{n})(1-2\varepsilon)\cdot\text{OPT} > (1-3\varepsilon)\cdot\text{OPT}$ for large enough n. □

4.4.3 Random Order Lower Bound

Proof of Lemma 4.3.3. We invoke 4.3.2 with the following setting of variables. Let Y be a binary vector where $Y_i = 1$ for $i \in [k]$ and $Y_i = 0$ otherwise. (Y_i is the indicator for whether an item is a public item.) S represents the set of indices in the ε-prefix. Then, Y_S counts the number of public

items in the ε-prefix in a random ordering of the items. Clearly, $\mathbb{E}[\,]Y_S] = \varepsilon k$, $\|Y\|_\infty = 1$, and $\|Y\|_2 = \sqrt{k}$. Finally, set $\delta = 2e^{-\frac{\varepsilon k}{8}}$, or more so $\ln\left(\frac{2}{\delta}\right) = \frac{\varepsilon k}{8}$. Now, by 4.3.2, we have

$$\Pr\left[|Y_S - \varepsilon k| \geq \frac{2}{3}\cdot\frac{\varepsilon k}{8} + \sqrt{k}\cdot\sqrt{2\varepsilon\cdot\frac{\varepsilon k}{8}}\right] = \Pr\left[|Y_S - \varepsilon k| \geq \frac{7}{12}\cdot\varepsilon k\right] \leq 2e^{-\frac{\varepsilon k}{8}}.$$

and the lemma follows. □

Proof of Lemma 4.3.4. Fix a type of private item, say those of type-j, i.e. only agent j (for some $j \in [n-1]$) has unit value for this type of item while all other agents have 0 value. First, we bound the probability that no item of type-j appears in the ε-prefix. To do this, note that this probability can be written, using the chain rule for conditional probabilities, as the product (over i from 1 to εnk) of the probabilities of the ith item in the ε-prefix not being a type-j item under the condition that the first $i-1$ items were not type-j items either. Clearly, this probability, for any $i \leq \varepsilon nk$, is at least $1 - \frac{1}{(1-\varepsilon)n}$ since there are k items of type-j among at most $(1-\varepsilon)nk$ items overall after the conditioning. Thus, the probability that no item of type-j appears in the ε-prefix is at least

$$\left(1 - \frac{1}{(1-\varepsilon)n}\right)^{\varepsilon nk} \geq 4^{\frac{-\varepsilon}{1-\varepsilon}\cdot k} \text{ by choosing } n \geq \frac{2}{1-\varepsilon}.$$

Denote $p = 4^{\frac{-\varepsilon}{1-\varepsilon}\cdot k}$. Consider the events that at least one item of type-j appears in the ε-prefix. These events are negatively correlated and therefore, the probability that at least one item of type-j appears in the ε-prefix for every $j \in [n-1]$ is at most

$$(1-p)^{n-1} \leq \left(1 - 4^{\frac{-\varepsilon}{1-\varepsilon}\cdot k}\right)^{n-1} \leq e^{-4^{\frac{-\varepsilon}{1-\varepsilon}k}(n-1)}.$$

□

Having verified the two key lemmas, we proceed to complete the proof of our main algorithmic result.

Proof of Theorem 4.3.1. Let $k = \frac{1-\varepsilon}{2\varepsilon} \cdot \lg(n-1)$. Then,

$$(n-1) \cdot 4^{\frac{-\varepsilon}{1-\varepsilon}k} = (n-1) \cdot 2^{-\frac{2\varepsilon}{1-\varepsilon} \cdot \frac{1-\varepsilon}{2\varepsilon} \cdot \lg(n-1)} = (n-1) \cdot 2^{-\lg(n-1)} = 1,$$

and furthermore,

$$\frac{\varepsilon k}{8} = \frac{1-\varepsilon}{16} \cdot \lg(n-1).$$

Plugging these expressions into 4.3.3 and 4.3.4, we get that the probability of every private item type appearing in the ε-prefix is at most $\frac{1}{e}$ and the probability of fewer than $\frac{5\varepsilon}{12}$ fraction of public items appearing in the ε-prefix is at most $2e^{-\frac{1-\varepsilon}{16} \cdot \lg(n-1)}$. The latter probability can be further simplified to

$$2e^{-\frac{1-\varepsilon}{16} \cdot \lg(n-1)} = 2e^{-\frac{1-\varepsilon}{16} \cdot \frac{\ln(n-1)}{\ln 2}} = \left(\frac{2}{n-1}\right)^{\frac{1-\varepsilon}{16 \ln 2}} < \frac{1}{e} \text{ for large enough } n.$$

Using the union bound over these two events, we conclude that with probability at least $1 - \frac{2}{e}$, there is at least one missing private item type in the ε-prefix, and also at least $\frac{5\varepsilon}{12}$ fraction of private items appear in the ε-prefix. In this case, in the ε-prefix, the algorithm cannot distinguish between the private agent whose item type is missing and the public agent. Thus, in expectation (over the randomness of the algorithm), at least half the public items in the ε-prefix are assigned to the private agent whose item type is missing. As a consequence, at least $\frac{5\varepsilon}{24}$ fraction of the

public items are not allocated to the public agent in the entire algorithm, which means that the total valuation of the public agent is at most $\left(1 - \frac{5\varepsilon}{24}\right) k$. Thus, in order to guarantee

$$\text{ALG} \geq (1 - \gamma) \cdot \text{OPT},$$

we need $\gamma \geq \frac{5\varepsilon}{24}$, i.e., $\frac{1}{\varepsilon} \geq \frac{5}{24\gamma}$. This implies by the set value of k that

$$\text{OPT} = k = \frac{1-\varepsilon}{2\varepsilon} \cdot \lg(n-1) = \frac{\frac{1}{\varepsilon} - 1}{2} \cdot \lg(n-1) \geq \frac{\frac{5}{24\gamma} - 1}{2} \cdot \lg(n-1) \geq C \cdot \frac{\ln n}{\gamma} \text{ for some constant } C.$$

This completes the proof of 4.3.1.

□

4.5 Conclusions, Limitations & Future Works

In the current work, we have presented impossibility results for the online Santa Claus problem in adversarial and random order input models, as well as a near optimal algorithm for the random order setting. These results effectively address the necessary assumptions on the problem to obtain optimal online solutions, and furthermore obtain these results via simplistic algorithms. Since the MAXMIN objective function is the dual of MINMAX used in the makespan and load balancing literature, we hope that the impossibility results presented here can be carried over to address the remaining open questions regarding their tightest additive terms in competitive ratio analysis.

Chapter 5: Fairness and Efficiency in Online Class Matching

5.1 Introduction

The rapid advancement of technology and the widespread adoption of online platforms have revolutionized the way we interact, conduct business, and access services. From ride-sharing platforms [Banerjee and Johari, 2019] to online marketplaces [Mehta et al., 2007], these platforms connect users with a vast array of resources, creating unprecedented opportunities for dynamic resource allocation. However, efficiently matching supply with demand in such *online* environments poses significant challenges, necessitating the exploration of novel algorithms and strategies.

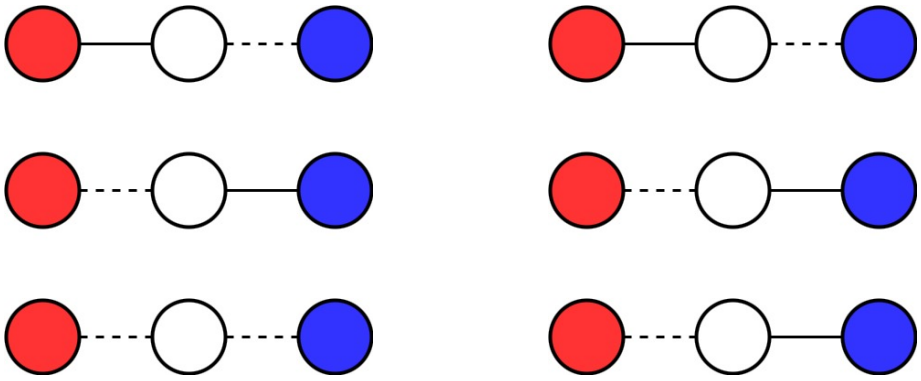
The fundamental online matching problem lies at the core of resource allocation in such platforms. Unlike traditional matching problems where the entire set of agents and resources are known in advance, the online matching problem involves making real-time decisions without complete information about future arrivals and requests. This inherent uncertainty and dynamic nature render traditional static matching algorithms inadequate, demanding the development of new techniques tailored specifically for online settings.

The objective of the online matching problem is to match as many arriving goods to static (offline) agents as possible in an efficient manner. The realization of this objective may vary depending on the specific application context. However, regardless of the objective, the challenge

lies in making immediate decisions while accounting for future arrivals and the scarcity of resources. The performance evaluation of algorithms designed for this problem is based on their competitive ratio, representing the worst-case approximation ratio between the size of the produced matching and the maximum possible size with complete hindsight. It is well known that the best deterministic algorithm can only achieve a $\frac{1}{2}$-approximation and the seminal work of [Karp et al., 1990] provides a randomized algorithm with a tight $(1 - \frac{1}{e})$-approximation guarantee.

To motivate the study of online matching under fairness constraints [Hosseini et al., 2024], we turn to the real-world challenges presented by modern internet economics and emerging marketplaces. These settings demand solutions that balance transparency with fairness, as emphasized in Moulin's "Fair Division in the Internet Age" [Moulin, 2019]. In various applications, such as allocating advertisement slots [Mehta et al., 2007], assigning packets in switch routing [Azar and Richter, 2004], distributing food donations [Mertzanidis et al., 2024], and matching riders to drivers in ridesharing platforms [Dickerson et al., 2021], items or services must be matched to agents immediately and irrevocably as they arrive. However, much of the existing work overlooks the need for fairness in matching decisions. Consider, for instance, a food bank that must allocate perishable food items upon arrival. Ensuring that these resources are distributed equitably across all communities is crucial to addressing fairness concerns in these high-stakes scenarios.

It is for precisely this reason that [Hosseini et al., 2024] initiate the study of *class fair matchings* where a set of items arriving online must be assigned to agents, who are partitioned into known classes, with the goal of achieving fairness among classes. In this problem, similar to online bipartite matching, agents either like an item (value 1) or do not like it (value 0), however our objective is to ensure equitable treatment of different classes. We refer to the standard



(a) CEF Matching (b) NW Matching

Figure 5.1: Examples of class envy-free (CEF) and non-wasteful (NW) matchings where bolded lines indicate a matching. Red nodes indicate agents in the first class, blue nodes indicate agents in the second class, and white nodes indicate items.

unfair objective that maximizes the number of matched agents as the utilitarian social welfare (USW). When considering classes, the notion of class envy-freeness (CEF) ensures that no class of agents can enhance their overall value by obtaining the items allocated to another class, even if the items are optimally distributed within their own class. It is important to note that in cases involving indivisible items, a class envy-free matching may not always be possible (see Figure 5.1). Consequently, our research mainly focuses on addressing the central *unresolved question* in the online class fair matching problem posed by [Hosseini et al., 2024]:

Problem 5.1.1. *Can a randomized algorithm for matching indivisible items achieve any reasonable CEF approximation together with either non-wastefulness or a USW approximation?*

5.1.1 Our Results

Our work mainly focuses on *randomized* algorithms for matching *indivisible* items in a fair manner, subject to the varying definitions of fairness adapted from the fair division literature. Notably, we provide the first non-wasteful algorithm that simultaneously obtains approximate

class fairness guarantees in expectation, resolving Open Problem 5.1.1 in an affirmative manner. This algorithm is a natural random matching procedure that exhibits notable constant factor approximations in spite of its simplicity. Our main algorithmic result is stated formally as follows.

Theorem 5.1.2 (Randomized algorithm; informal)**.** *For a randomized matching of indivisible items, the* RANDOM *algorithm satisfies non-wastefulness,* $\frac{1}{2}$*-CEF,* $\frac{1}{2}$*-CPROP, and* $\frac{1}{2}$*-USW.*

We highlight that the analysis of the various approximate fairness guarantees is highly non-trivial due to the non-additive nature of the objective functions, to be discussed more formally in Section 5.3. In exploring the tightness of our algorithmic guarantees, we additionally construct an upper bound adversarial input that demonstrates the limits on achievable fairness in this problem setting.

Theorem 5.1.3 (Indivisible CEF upper bound; Informal)**.** *Any non-wasteful (possibly randomized) algorithm cannot achieve an* α*-CEF guarantee for* $\alpha > \frac{e^2-1}{e^2+1} \approx 0.761$.

This upper bound construction builds upon the results of [Karp et al., 1990] to demonstrate that CEF cannot be achieved and will be presented in Section 5.4.1. We additionally note that while our results show a deviation in the guarantees from traditional fair division problems, certain properties nicely translate to our problem setting—we expound on this fact in Section 5.6 with a comprehensive discussion on the connections between *class* Nash welfare and CEF1.

In Section 5.4.2, we additionally provide a strengthened upper bound for the *divisible* matching setting through a careful input construction which further resolves an open problem left by [Hosseini et al., 2024].

Theorem 5.1.4 (Divisible CEF upper bound; Informal)**.** *No deterministic algorithm for divisible matching can achieve* β*-CEF for any* $\beta > 0.677$.

Indivis.	USW	CEF	CPROP	**Divis.**	USW	CEF	CPROP
Alg.	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	Alg.	$\frac{1}{2}^{\dagger}$	$1-\frac{1}{e}^{\dagger}$	$1-\frac{1}{e}^{\dagger}$
Bound	$1-\frac{1}{e}^{\ddagger}$	$\frac{e^2-1}{e^2+1}$	$1-\frac{1}{e}$ [1]	Bound	$1-\frac{1}{e}^{\ddagger}$	0.67	$1-\frac{1}{e}^{\dagger}$

Table 5.1: The summary of our results on randomized algorithms. Each algorithm achieves its three guarantees simultaneously, while the upper bound holds for any algorithm, separately for each guarantee. Results from prior works are denoted with † for [Hosseini et al., 2024] or ‡ for [Karp et al., 1990].

Finally, in an effort to further quantify the inherent gap between fair and unfair solutions to the online matching problem, we conclude the paper with a definition for the "price of fairness" which is intuitively the necessary trade-off between an optimal and a fair matching. We demonstrate that as we strive for a higher level of fairness in the approximation of the solution, the objective of maximizing utilitarian social welfare (USW) deteriorates, exhibiting an inverse proportionality relationship.

Theorem 5.1.5 (Price of fairness; Informal)**.** *For any $\varepsilon > 0$, there exists a problem instance such that no (possibly randomized) online algorithm that guarantees α-CEF can achieve USW larger than $\frac{1}{1+\alpha} + \varepsilon$.*

The proofs for each result are sketched in the corresponding sections, with full analysis presented in Section 5.7.

5.1.2 Related Work

Online Matching. For an extensive exploration of the vast literature on online matching, we refer readers to [Mehta et al., 2013], while summarizing key findings relevant to this paper. The seminal work by [Karp et al., 1990] introduces the RANKING algorithm which, in our problem

[1] This fact trivially holds when considering the upper-triangular construction of [Karp et al., 1990] and $k = 1$ in our setting.

setting, corresponds to a randomized algorithm for matching indivisible items that achieves a utilitarian social welfare (USW) approximation of $1 - \frac{1}{e}$. In the fractional matching domain, an identical result is achieved with a deterministic algorithm [Kalyanasundaram and Pruhs, 2000]. The literature further explores randomized input models of online matching problems to surpass this well-known $1-\frac{1}{e}$ barrier. For online vertices that arrive in a random order (reducing the power of an adversarial input) [Mahdian and Yan, 2011] and [Karande et al., 2011] demonstrate that the competitive ratio of the Ranking algorithm falls between 0.696 and 0.727. Moreover, [Huang et al., 2019] propose a variant of Ranking that surpasses the $1-\frac{1}{e}$ barrier in vertex-weighted online matching under random-order arrivals, with a further improvement to 0.668 presented by [Jin and Williamson, 2021]. In stochastic matching, where items are drawn from a known distribution, the best-known competitive ratios for unweighted and vertex-weighted online stochastic matching are 0.711 and 0.700, respectively [Feldman et al., 2009a, Huang and Shu, 2021].

Fair Division. In the offline setting, envy-freeness (EF) and proportionality (PROP), along with their approximations, are commonly employed as criterions of fairness in the allocation of items to agents. For divisible items, an allocation which is envy-free and pareto optimal (PO) always exists [Varian, 1973] and can be computed efficiently when agent valuation functions are additive [Eisenberg and Gale, 1959]. For indivisible items, such allocations are not guaranteed to exist. As such, two relaxations are commonly studied: envy-freeness up to one item (EF1) [Lipton et al., 2004] and maximin share fairness (MMS) [Budish, 2011]. An allocation that satisfies EF1 is guaranteed to exist when valuations are monotone, and are further PO when agents have additive valuations [Caragiannis et al., 2019b]. However, the existence of MMS allocations is not guaranteed, even for additive valuations. Nevertheless, there are various polynomial time ap-

proximation algorithms [Garg and Taki, 2020, Ghodsi et al., 2018, Hosseini et al., 2022, Hosseini and Searns, 2021, Kurokawa et al., 2018]. In our fair matching problem, we effectively compute an online allocation that satisfies the aforementioned fairness criteria by treating each class as an agent within the system.

Fair Online Matching. Most closely related to our work is the preliminary work of [Hosseini et al., 2024] that introduces the online class fair problem. This paper provides the initial results for approximate guarantees on the objectives of class envy-freeness (CEF), class proportionality (CPROP), class maximin share (CMMS), and utilitarian social welfare (USW). The authors' contributions offer nearly tight results for deterministic algorithms in both the context of indivisible and divisible item matching. While the authors achieve a 0.593-CPROP approximation guarantee for indivisible matching using a randomized algorithm, they do not address the challenge of simultaneously achieving a class envy-freeness (CEF) guarantee while ensuring non-wastefulness. This crucial open problem serves as the primary focus of our study. Furthermore, we explore various impossibility results for algorithms that align with the fairness-agnostic online matching literature, shedding light on the limitations and constraints of such approaches. While many other works examine online fair division [Aleksandrov et al., 2015, Benade et al., 2018, Gorokh et al., 2020, Walsh, 2011, Zeng and Psomas, 2020], the majority restrict attention to additive valuations, rendering their techniques inapplicable to the matching setting. Finally, the recent work by [Yokoyama and Igarashi, 2023] proposes a non-wasteful algorithm which guarantees CEF with high probability when the number of agents approach infinity.

Price of Fairness. The first set of results on the Price of Fairness (PoF) traces back to [Bertsimas et al., 2011] and [Caragiannis et al., 2012]. [Bertsimas et al., 2011] analyze the upper bound

on the utility loss (specifically, egalitarian social welfare) incurred by fairness notions such as proportional and max-min fairness in the allocation of divisible goods. A key takeaway from their results is that for a small number of agents, the PoF remains relatively low; for example, for two agents, the PoF for proportional fairness is at most 8.6%, and for max-min fairness, it is 11.1%. [Caragiannis et al., 2012] further extend these results by examining fairness notions like proportionality, envy-freeness, and equitability for both divisible and indivisible goods and chores. However, as [Bei et al., 2021] highlight, a significant limitation in the indivisible setting is that the guarantees do not hold for every problem instance, as the results are not framed as a worst-case analysis. To address this, [Bei et al., 2021] investigate PoF under worst-case scenarios for various fairness criteria, including Nash social welfare, envy-freeness up to one good, balancedness, and egalitarian social welfare. It is important to note that these results do not directly apply to our setting, as the notion of class envy-freeness is not equivalent to any of the properties discussed above.

5.2 Model

For $t \in \mathbb{N}$, define $[t] = \{1, ..., t\}$. Consider a bipartite graph $G = (\mathcal{N}, \mathcal{M}, \mathcal{E})$ where $|\mathcal{N}| = n$ represents a set of vertices henceforth referred to as "agents", $|\mathcal{M}| = m$ a set of vertices called "items", and $\mathcal{E}$ the set of incident edges. We say that $a \in \mathcal{N}$ likes item $o \in \mathcal{M}$ if the two are adjacent in G, i.e., $(a, o) \in \mathcal{E}$. The set of agents $\mathcal{N}$ is partitioned into k (known) classes $N_1, ..., N_k$ such that $N_i \cap N_j = \emptyset$ for all $i \neq j$ and $\bigcup_{i=1}^{k} N_i = \mathcal{N}$. We slightly abuse notation and call class N_i, class i.

Matching. We denote by the matrix $X = (x_{a,o})_{a \in \mathcal{N}, o \in \mathcal{M}} \in \{0, 1\}^{n \times n}$ a matching, where each

$x_{a,o}$ indicates if item o is matched to agent a. For divisible matchings, we replace $\{0,1\}$ by $[0,1]$. Given such a matching, we refer to an agent as *saturated* when $\sum_{o\in\mathcal{M}} x_{a,o} = 1$ and an item as *assigned* if $\sum_{a\in\mathcal{N}} x_{a,o} = 1$.

For some matching X, we let $Y(X) = \left(\sum_{a\in N_i} x_{a,o}\right)_{i\in[k],o\in\mathcal{M}}$ be the matrix containing the items assigned to agents within each class. Further let $Y_i(X)$ denote the row of $Y(X)$ corresponding to class i. More specifically, this is the set of items matched to agents in class i:

$$Y_i(X) = \{o \in \mathcal{M} : x_{a,o} = 1 \text{ for some } a \in N_i\}.$$

Class Valuations. For agent $a \in \mathcal{N}$, the value of a matching X is given by

$$V_a(X) = \sum_{o\in\mathcal{M}:(a,o)\in\mathcal{E}} x_{a,o}$$

and we slightly abuse notation by defining the value of class i from matching X to be $V_i(X) = \sum_{a\in N_i} V_a(X)$. This is the so-called *utilitarian social welfare* (USW) of the matching for the agents in class i, and is equivalent to the standard matching size objective in the online bipartite matching literature.

Given a vector $\mathbf{y} = (y_o)_{o\in\mathcal{M}} \in \{0,1\}^m$ representing the allocation of different items, the *optimistic valuation*, $V_i^*(\mathbf{y})$, of class i for $\mathbf{y}$ is the size of the maximum matching between the agents of N_i and the items of $\mathbf{y}$. V_i^* is equivalent to the maximum size of the integral matching between the items and agents in N_i which can be computed with a standard LP—we defer the reader to [Hosseini et al., 2024] for more exposition on this definition. We emphasize that the optimistic valuation function is *subadditive*, and not additive as is a standard assumption leveraged

in the fair division literature to obtain many algorithmic guarantees. As demonstrated in [Hosseini et al., 2024], it is exactly this functional property that prevents any deterministic algorithm for indivisible matchings from being non-wasteful and CEF1. Moreover, this aspect of the problem instance will necessarily make the analysis of our algorithmic guarantees and impossibility results non-trivial as compared to their additive counterparts.

5.2.1 Definitions of Fairness

The aim in the present work is to distribute the arriving goods among classes in a way that respects certain fairness criteria or principles. The commonly studied fairness criteria that we use here are envy-freeness [Foley, 1967], and proportionality [Steinhaus, 1948].

For envy-freeness, we compare the value of $V_i(X)$ for the matched items to class i and $V_i^*(Y_j(X))$, the optimistic value of class j's matching according to i. Note that the optimistic valuation is necessarily larger than what could be obtained in the online model, so this is a particularly strong notion of fairness.

Definition 5.2.1 (Class Envy-Freeness)**.** A matching X is α-**class envy-free** (α-CEF) if for all classes $i, j \in [k], V_i(X) \geq \alpha \cdot V_i^*(Y_j(X))$. For $\alpha = 1$, we simply call the matching **class envy-free** (CEF).

In general, CEF allocations cannot be guaranteed for indivisible matchings (ie. one item arrives to be distributed across two classes). We thus also consider the relaxed notion of class envy-freeness up to one item, consistent with the EF1 notion introduced by [Lipton et al., 2004].

Definition 5.2.2 (Class Envy-Freeness Up to One Item)**.** A matching X is α-**class envy-free up to one item** (α-CEF1) if for every pair of classes $i, j \in [k]$, either $Y_j(X) = \emptyset$ or there exists an

item $o \in Y_j(X)$ such that $V_i(X) \geq \alpha \cdot V_i^*(Y_j(X) \setminus \{o\})$. When $\alpha = 1$, we simply refer to the matching as **class envy-free up to one item** (CEF1).

At the class level, the *proportional share* of class i is defined as

$$\text{prop}_i = \max_{X \in \mathcal{I}} \min_{j \in [k]} V_i^*(Y_j(X))$$

where $\mathcal{I}$ is the set of (possibly divisible) matchings of the set of items M to the set of agents N.

Definition 5.2.3 (Class Proportional Fairness)**.** We say that a matching X is α-**class proportional** (α-CPROP) if for every class $i \in [k], V_i(X) \geq \alpha \cdot \text{prop}_i$. When $\alpha = 1$, we simply call this **class proportional** (CPROP).

We highlight that [Hosseini et al., 2024] demonstrate that any algorithm which assigns deterministically at the class level or within classes must be at best $\frac{1}{2}$-CPROP. Therefore, we must introduce randomness at both stages to surpass the performance of a deterministic algorithm. This further motivates the present studies' emphasis on randomized algorithms.

5.2.2 Definitions of Efficiency.

We consider two notions of efficiency. The first, non-wastefulness, ensures that no item is discarded if a matching is possible. In our integral assignment setting, non-wastefulness corresponds to a *maximal* matching.

Definition 5.2.4 (Non-Wastefulness)**.** We say that a matching X is **non-wasteful** (NW) if there is no pair of agent a and item o such that a likes o, a is not saturated, and o is not fully assigned.

The second efficiency measure is utilitarian social welfare which quantifies the size of the resultant matching. This is consistent with the classical objective for online matching when ignoring considerations of fairness.

Definition 5.2.5 (Utilitarian Social Welfare)**.** The **utilitarian social welfare** (USW) of a matching is given by $\text{USW}(X) = \sum_{a\in\mathcal{N}} \sum_{o\in\mathcal{M}:(a,o)\in\mathcal{E}} x_{a,o}$. We say that a matching is α-USW if $\text{USW}(X) \geq \alpha \cdot \text{USW}(X^*)$ for all matchings X^*. When $\alpha = 1$, we refer to X as the USW-optimal matching.

It is well-known that maximal matchings (both divisible and indivisible) are a at least a $\frac{1}{2}$-approximation to the maximum. The proof of this fact is standard in the literature for maximum and maximal matchings, but is included here for completeness and to ensure maximal clarity of our notation.

Proposition 5.2.6. *Every non-wasteful matching is $\frac{1}{2}$-USW.*

Proof. Let X^* be a matching that maximizes the USW objective, and let X be any non-wasteful matching in the same instance. By definition of non-wastefulness above, we must have that each edge in the graph G has at least one end point included in the matching, ie. for every $(a, o) \in \mathcal{E}$, $\sum_{o'\in\mathcal{M}} x_{a,o'} = 1$ or $\sum_{a'\in\mathcal{N}} x_{a',o} = 1$. Therefore, we have

$$\begin{aligned}
\text{USW}(X^*) &= \sum_{a\in\mathcal{N}} \sum_{o\in\mathcal{M}} x^*_{a,o} \\
&\leq \sum_{(a,o):x^*_{a,o}=1} \left(\sum_{a'\in\mathcal{N}} x_{a',o} + \sum_{o'\in\mathcal{M}} x_{a,o'} \right) \\
&\leq \sum_{a\in\mathcal{N}} \sum_{o'\in\mathcal{M}} x_{a,o'} + \sum_{a'\in\mathcal{N}} \sum_{o\in\mathcal{M}} x_{a',o} \\
&= 2 \cdot \text{USW}(X)
\end{aligned}$$

where the first inequality comes from the fact that $x^*_{(a,o)} = 1$ implies at least one of the end points for each (a, o) included in the maximum maching must be in the non-wasteful matching. By the summation properties on edges in a non-wasteful matching we obtain the final equality and, therefore, a $\frac{1}{2}$-USW approximation. □

5.2.3 Online Model

In the online setting, the items in $\mathcal{M}$ arrive one-by-one in an arbitrary order. We refer to the step in which item $o \in \mathcal{M}$ arrives as step o. Upon the arrival of item o, the incident edges, (a, o), to the agents in $a \in \mathcal{N}$ are revealed from G. At this point, the algorithm must make an irrevocable decision to match the item one of the agents in $\mathcal{N}$ who is not currently saturated (ie. not already included in the matching). We examine both deterministic (for analytic purposes) and randomized algorithms to construct these matchings.

In the online setting we define the online fairness metrics using the standard notion of a competitive ratio as our approximation factor as follows.

Definition 5.2.7. For $\alpha \in (0, 1]$, a deterministic online algorithm for matching items is α-CEF (resp., CEF1, CPROP, USW, or NW) if it produces an α-CEF (resp., CEF1, CPROP, USW, or NW) matching after all items have arrived.

For randomized algorithms, we require that all fairness constraints hold in expectation after all items have arrived. However, we often demonstrate that our algorithms maintain this invariant in expectation at each iteration.

5.3 Randomized Algorithms

We here present our randomized algorithm for constructing a matching that achieves simultaneous guarantees for the CEF and CPROP fairness objectives, as well as non-wasteful efficiency. While wastefulness makes the fairness objectives somewhat trivial to obtain, our enforced non-wasteful condition showcases the complexity of maintaining a fair matching. We begin by analyzing our algorithm and later use this procedure to validate the impossibility result on the α-CEF guarantee of any algorithm.

The algorithm is a simple variant on a completely random matching procedure to ensure non-wasteful efficiency. Upon the arrival of an item o, it is revealed which classes $i \in [k]$ have a currently unmatched agent, a, that likes the item (ie. $(a, o) \in \mathcal{E}$ and $x_a = 0$). Over this set of classes, we select one at random and then randomly assign the item to an unmatched agent that likes the item within this class. Though this nested randomization appears obvious, the proof of its near optimal fairness guarantees requires a nontrivial analysis. The following theorem establishes the approximate fairness and efficiency guarantees of our algorithm—the pseudocode is provided in Algorithm 7.

Algorithm 7 RANDOM

for $o \in M$ **do**
  $S_o \leftarrow \emptyset$
  for $i \in [k]$ **do**
    if $\exists a \in N_i$ s.t. $(a, o) \in E$ and $x_a = 0$ **then**
      $S_o \leftarrow S_o \cup \{i\}$
    end if
  end for
  Pick an $i \in S_o$ uniformly at random
  Pick an $a \in N_i$ with $(a, o) \in E$ and $x_a = 0$ uniformly at random
  Set $x_{a,o} = 1$
end for

Theorem 5.3.1 (Theorem 5.1.2 (Formal))**.** *For randomized matching of indivisible items, Algorithm 7 satisfies non-wastefulness, $\frac{1}{2}$-CEF, $\frac{1}{2}$-CPROP, and $\frac{1}{2}$-USW*

Proof Sketch. The non-wastefulness of the algorithm is direct from its definition: each arriving item is allocated to an unsaturated agent that likes the item. The USW approximate guarantee then follows from Proposition 5.2.6.

For the CEF objective, we invoke a novel proof technique specific to the challenging analysis of optimistic valuations used throughout the fair matching objective framework. Specifically, we show that for any two distinct classes i, j, the expected value $\mathbb{E}[V_i(X)] \geq \frac{1}{2} \cdot \mathbb{E}[V_i^*(Y_j(X))]$ by introducing dummy items to analyze the value of some augmented input set A_i, proving that $V_i^*(A_i) \leq 2 \cdot V_i(X)$. With this augmented set, we more readily obtain the desired approximate guarantees and demonstrate that the optimal solution on A_i dominates the true solution on $Y_j(X)$. Combining this fact with the following lemma relating the expected values, we establish the $\frac{1}{2}$-CEF guarantee.

Lemma 5.3.2. *For any two distinct classes $i, j \in [k]$, $\mathbb{E}[V_i^*(A_i)] \geq \mathbb{E}[V_i^*(Y_j(X))]$ where expectations are taken over the randomness of Algorithm 7.*

We suspect this novel proof technique will be useful in follow up works that examine similar, nonadditive, solution concepts—a key problem identified by the preliminary work of [Hosseini et al., 2024].

Lastly, for the CPROP objective, the analysis relaxes the Equal-Filling-OCS approach by using a simpler independent rounding method. More concretely, for an arriving item $o \in \mathcal{M}$ we construct the vector $(x_{a,o})_{a \in \mathcal{N}}$ where each entry is the corresponding probability of an agent being matched to the given item under the RANDOM algorithm. After all items have arrived, each

agent a is matched to an item with probability

$$1 - \prod_{o \in \mathcal{M}} (1 - x_{a,o}) \geq 1 - \exp\left(-\sum_{o \in \mathcal{M}} x_{a,o}\right)$$

and by integrating according to this density function over the agents in each class and comparing to the upper bound on the prop_i value, we obtain the desired approximation ratio. □

We here highlight that while the more sophisticated OCS rounding scheme and, as a result, the Equal-Filling-OCS algorithm yields a stronger 0.593-CPROP guarantee, the algorithm does not lend itself to analysis of the CEF objective (it remains an open problem to obtain any such approximation for the algorithm). Thus, although our algorithm is slightly weaker with respect the CPROP fairness definition, our simple algorithm further gives a $\frac{1}{2}$-CEF approximation while maintaining non-wastefulness.

5.4 Improved CEF Upper Bounds

In this section, we provide improved CEF upper bounds of 0.761 for any randomized indivisible and 0.677 for any deterministic divisible algorithm.

5.4.1 Indivisible setting

Our upper bound for the indivisible setting showcases the near tightness of our algorithmic guarantees proven in Section 5.3.[2]

The seminal paper of [Karp et al., 1990] showed that no online algorithm can get a com-

[2]We note that, trivially, an upper bound of $1 - \frac{1}{e}$ exists for the USW objective by the classic result of [Karp et al., 1990]. This bound persists for CPROP by considering the problem instance where $k = 1$.

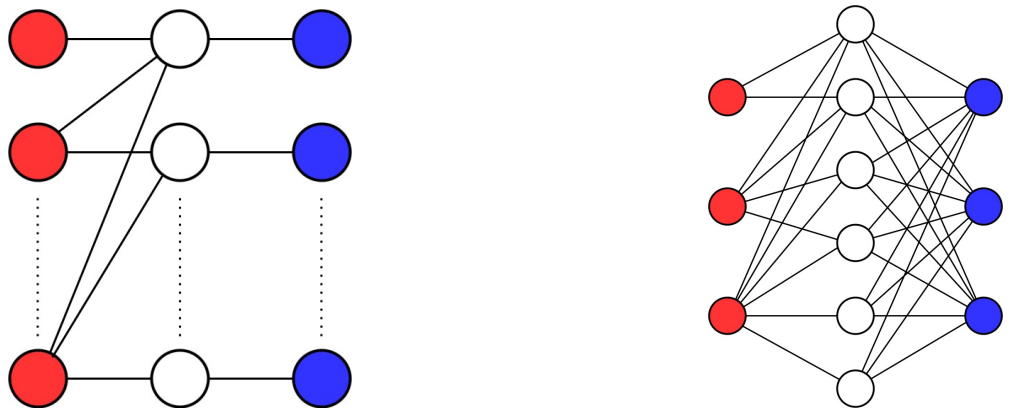
petitive ratio better than $1 - \frac{1}{e}$ for the USW objective by an "upper triangular graph" (the graph whose adjacency matrix is upper triangular) construction. In this graph, there are n arriving items and n agents. The first item to arrive has an edge to all n agents, the second has an edge to $n - 1$ of the agents, the third to only $n - 2$ of them, and so on.

We will proceed with demonstrating that by an extension on the result of [Karp et al., 1990], we can upper bound the α-CEF guarantee of any randomized algorithm. Specifically, we prove the following theorem:

Theorem 5.4.1 (Theorem 5.1.3 (Formal))**.** *No randomized online algorithm for matching indivisible items can achieve an α-CEF guarantee for any $\alpha > \left(\frac{e^2-1}{e^2+1}\right)$ and non-wastefulness.*

Our construction for CEF impossibility extends the problem instance of [Karp et al., 1990] to include a second class which can be uniquely matched to each arriving item (See Figure 5.2a for an illustration of this instance). We proceed to show that this instance admits at best a $\frac{e^2-1}{e^2+1}$ approximation to the CEF objective by first showing that the RANDOM algorithm admits this approximation factor in expectation and subsequently showing that no algorithm can do better on the given instance, thus bounding the performance of any randomized algorithm in general.

Most crucial to the argument that bounds α is the computation of a "stopping time" after which no further items can be matched to N_1 since all potential agents will have been previously saturated in a suboptimal manner—a result of the ambiguity in matching from the upper triangular instance and randomization of our algorithm. After this stopping time, by non-wastefulness, all items will be matched to the second class. This necessarily results in an unfair distribution of items and by deriving the stopping time as a fraction of the number of items, we obtain our upper bound approximation.



(a) (b)

Figure 5.2: Impossibility constructions for the upper bound results of Theorems 5.4.1 and 5.1.4. (a) the indivisible setting construction for an at most $\left(\frac{e^2-1}{e^2+1}\right)$-CEF approximation, (b) the divisible setting construction for an at most 0.677-CEF approximation. Node order arrival is depicted from top to bottom.

5.4.2 Divisible setting

In this section, we improve upon the established bounds of [Hosseini et al., 2024] and move closer towards resolving the open problem on what is the best achievable α-CEF guarantee for non-wasteful divisible matchings through a novel input construction. Prior to this work, the best known upper bound bound was $\frac{3}{4}$ with an algorithm that guarantees at least $1 - \frac{1}{e}$. Our improved bound of 0.677 is thus nearly tight. Note that our CEF upper bound is subject to non-wastefulness because an algorithm can trivially achieve fairness on its own by throwing away every item.

Theorem 5.4.2 (Theorem 5.1.4 (Formal))**.** *No non-wasteful deterministic algorithm for matching divisible items can achieve β-CEF for any $\beta > 0.677$.*

Proof Sketch. We construct an adversarial instance where achieving β-CEF is impossible for any $\beta > 0.677$. The instance consists of two classes, each containing n agents, and an input stream of $2n$ items. In the first class, the connectivity structure follows a hierarchical pattern: items 1 and

2 are connected to all agents. For each $i \geq 1$, items $2i+1$ and $2i+2$ inherit the connections of items $2i-1$ and $2i$, except that one arbitrary agent is removed from the connections. As a result, at each step, items $2i-1$ and $2i$ are connected to precisely $n-i+1$ agents. The second class follows a simpler structure, where every agent is connected to every item, forming a complete bipartite graph between agents and items. For clarity, Figure 5.2b illustrates the full construction for the case $n = 3$.

To verify the result for the given instance, assume there is an algorithm that guarantees α-CEF. The proof then follows from the synthesis of two key facts: (i) any α-CEF algorithm should distribute items equally among agents within N_1 for this input instance to maximize their saturation (Lemma 5.7.6), and (ii) that any such algorithm must further divide arriving items between the two classes such that N_1 receives $\frac{1+\alpha}{2}$ and N_2 receives $\frac{1-\alpha}{2}$. This ratio is optimal against an adversarial input, ensuring neither class is overly envious (Lemma 5.7.7).

The first result is proven by the nature of a non-wasteful online algorithm, and the second is achieved by induction: assume the α-CEF guarantee holds up to step $t-1$. For step t, if the item is not allocated according to the prescribed ratio, an adversary can force a violation of the α-CEF guarantee. Thus, maintaining the ratio ensures the guarantee persists.

The theorem is finally proven by bounding the matching size for each class given the two above facts. We can ultimately determine the maximum value of α that maintains the α-CEF guarantee, concluding that $\alpha \leq 0.677$. □

5.5 Price of Fairness

Stemming from the highly influential work of [Karp et al., 1990], it is well-known that no algorithm for the online matching problem can achieve an approximation better than $1 - \frac{1}{e}$ to the maximal matching objective (USW in our context). Moreover, by the result of Theorem 5.4.1, we demonstrate that a comparable approximation bound persists for the CEF objective. It is, thus, only natural to explore the following question: *is it possible to achieve both an optimal CEF and USW approximation simultaneously?*

We here address this question with an impossibility result that provides an initial trade-off between the fairness of a solution and its optimality with respect to the (unfair) USW objective—a relationship we refer to as the *price of fairness* for online matching. More formally, our result is as follows.

Theorem 5.5.1 (Theorem 5.1.5 (Formal))**.** *For any $\varepsilon > 0$, there exists a problem instance such that no (possibly randomized) non-wasteful online algorithm with an α-CEF guarantee can achieve an approximation to the USW objective greater than $\frac{1}{1+\alpha} + \varepsilon$.*

Proof Sketch. The proof proceeds by considering an instance with $k - 1$ classes of q agents, as well as a k-th class comprised of $q(k - 1)$ agents. The adversarial input stream consists of two phases. In the first phase, $p(k - 1) + q$ items arrive, each of which has incident edges to every agent in the graph, whereas in the second phase, $k-1$ groups of items arriving sequentially where the i-th such group consists of q items with edges to all the agents in class i.

In this instance, we first show that any non-wasteful α-CEF algorithm should allocate at least $p(k-1)$ items to the class $1, 2, \ldots, k-1$. Then, since these items cannot be further matched

to class $1, \ldots, k-1$ in the second phase, we can upper bound the utilitarian social welfare at the end of the second phase by $p(k-1) + qk - p(k-1) = qk$. Observing that the offline optimal solution in hindsight allocates all items in the first phase to class k, while allocating the remaining q items specific to each class to their corresponding class in the second phase, the optimal USW is $p(k-1) + q + q(k-1)$. The proof follows from selecting proper parameters for p and q. □

With this novel price of fairness result, we observe that if $\alpha > \frac{1}{e-1} \approx 0.582$, then our result for α-CEF implies a USW guarantee that is strictly smaller than $1 - \frac{1}{e}$, the well known bound by [Karp et al., 1990]. Thus, USW must be sacrificed to achieve fairness. Further, if there exists any algorithm that achieves the 0.761-CEF guaranteed by Theorem 5.1.3, it would necessarily have USW smaller than 0.568 (larger than 0.5 by maximality), which is considerably smaller than the $1 - \frac{1}{e} \approx 0.632$ by [Karp et al., 1990].

5.6 Class Efficiency & Nash Social Welfare

We lastly extend the classical notion of Nash social welfare (NSW) to the class fair matching problem setting and demonstrate its intersection with the notion of CEF1. Concretely, we demonstrate that the former implies the latter, but a reverse implication is not guaranteed.

Definition 5.6.1 (Class Nash Social Welfare)**.** The class Nash social welfare (*CNSW*) of a matching X is defined by

$$\text{CNSW(X)} = \left(\prod_{i=1}^{k} V_i(X) \right)^{1/k}.$$

We say that a matching is CMNW if it maximizes CNSW.

The NSW has been typically viewed as a balance between fair and efficient allocations of

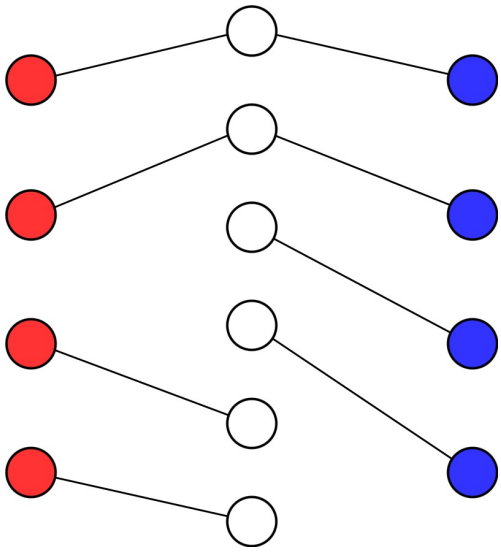


Figure 5.3: Hardness instance for Theorem 5.6.3.

items to a set of agets. Most recently, [Yuen and Suksompong, 2023] showed the NSW objective is the only welfarist function of agent valuations whose maximization leads to allocations that are EF1 and PO. In spite of its tremendous value, it is also widely known that maximizing the objective is generally hard even to approximate within polynomial time without strong problem assumptions [Barman et al., 2018]. We here demonstrate that, given such a maximizing matching for the CNSW objective, we further obtain the CEF1 guarantee in line with the fair division literature. For a more complete discussion on the NSW objectives role in fair division, we defer the reader to [Amanatidis et al., 2023].

Theorem 5.6.2. *A* CMNW *matching satisfies non-wastefulness and is* CEF1.

We further demonstrate that although the CMNW matching provides the favorable CEF1 fairness guarantee, the opposite relationship is not necessarily ensured. The following theorem formalizes this notion that CMNW is a strictly stronger property than CEF, which itself is stronger than CEF1.

Theorem 5.6.3. *A non-wasteful CEF matching is not guaranteed to maximize CNSW.*

This result is due to an adversarial input construction on $m = 6$ items and two classes of 4 agents each, as depicted in Figure 5.3.

5.7 Proofs

We here present the proofs of all theoretical results.

5.7.1 Randomized Algorithm Guarantees

Proof of Theorem 5.3.1. The non-wastefulness of the algorithm follows immediately from its definition: at each step, the arriving item is allocated to an unsaturated agent that likes the item. Also, USW immediately follows from Proposition 5.2.6. Thus, in what follows, we focus on proving CEF and CPROP guarantees of the algorithm.

CEF. Let X be the matching constructed by the randomized algorithm after the arrival of all items in $\mathcal{M}$. Consider any two distinct classes $i, j \in [k]$. We seek to prove that

$$\mathbb{E}\,[\,V_i(X)\,] \geq \frac{1}{2} \cdot \mathbb{E}\,[\,V_i^*(Y_j(X))\,].$$

According to our algorithm, for any item o liked by at least one agent in the class i, the probability of allocating this item to i is at least that of allocating to another class j unless, at step o, all agents who like the arriving item are already saturated.

For ease of analysis, we consider a simple upper bounding scheme that leverages the above fact. Specifically, upon the arrival of item o that is liked by at least one agent in i but all such agents are saturated, we create a dummy item $\overline{o}$ that is identical to o and add it to the set of matched items to class i. We therefore augment the set X_i to include these dummy items which

we denote as the set $A_i \supseteq X_i$. Using this, we prove the following claim.

Claim 5.7.1. *The optimistic value of set A_i to class i, denoted as $V_i^*(A_i)$, is at most twice the size of its matching in X. More formally:*

$$V_i^*(A_i) \leq 2 \cdot V_i(X).$$

Proof. We first observe that if $A_i = X_i$, then $V_i^*(A_i) = V_i^*(X)$ and since the optimistic value constructs a maximal matching, the claim holds by application of Proposition 5.2.6.

We therefore assume that $X_i \subsetneq A_i$. By the order of arrival of items, we must have that $V_i(X) = V_i(A_i)$ since the items in $A_i \setminus X_i$ can only be matched to agents that are currently saturated. However, under the optimistic valuation, the matching can only increase in size, i.e., $V_i^*(X) \leq V_i^*(A_i)$. Again, by application of Proposition 5.2.6 and the maximality of the optimistic value of A_i, we must have that

$$V_i^*(A_i) \leq 2 \cdot V_i(A_i) = 2 \cdot V_i(X)$$

completing the proof of the claim. □

The combination of Lemma 5.3.2 with Claim 5.7.1, we have that

$$\mathbb{E}\left[\, V_i(X) \,\right] \geq \frac{1}{2} \cdot \mathbb{E}\left[\, V_i^*(A_i) \,\right] \geq \frac{1}{2} \cdot \mathbb{E}\left[\, V_i^*(Y_j(X)) \,\right]$$

thus, we have the $\frac{1}{2}$-CEF guarantee and have proven the theorem. It therefore remains to prove Lemma 5.3.2.

Proof of Lemma 5.3.2. To prove the lemma, we need to verify two essential claims.

Claim 5.7.2. *For an arbitrary item, o, liked by an agent in class $i \in [k]$, the probability that o (or its copy) is in A_i is greater than or equal to the probability that $o \in Y_j(X)$ for $i \neq j \in [k]$.*

Proof. If there exists an agent in class i that likes o and is unsaturated at the time of its arrival, then the item will be added to X_i with equal probability to any other class j containing such an agent, so the claim holds.

If instead all such agents in class i are saturated, then a copy of o will be added to A_i with probability 1. Thus, we have the claim. □

We now leverage this claim to prove a slightly stronger result which will ultimately yield the lemma result.

Claim 5.7.3. *For an arbitrary item o liked by an agent of class i, we must have that at least one of the following properties hold:*

1. *for $j \neq i \in [k]$, $\Pr[\, o \in Y_j(X)\,] = 0$*
2. *$\Pr[\, o \in A_i\,] = 1$*
3. *or $\Pr[\, o \in Y_j(X)\,] = \Pr[\, o \in A_i\,]$*

Proof. Upon the arrival of $o \in \mathcal{M}$, there are three possibilities for its irrevocable assignment. If there is no agent in j that likes o, we must have that $\Pr[\, o \in Y_j(X)\,] = 0$. Alternatively, if such an agent exists and is currently unsaturated but all the viable agents in i are saturated, we must have that o is added to A_i. Lastly, if both classes have unsaturated agents that like the item, then either $Y_j(X)$ or $X_i \subseteq A_i$ will receive the item with equal probability. Thus, we have the claim. □

We now have the necessary tools to argue about the expected optimistic values of the sets A_i and $Y_j(X)$ according to class i. Clearly, only items in $Y_j(X)$ that are liked by agents of class i contribute to the optimistic valuation. By the above claims, each such item is allocated to A_i with greater than or equal to the probability that they are allocated to $Y_j(X)$. We therefore have the result by linearity of expectations for these item assignments. □

CPROP. The analysis of our CPROP guarantee relies on a modification to that of Equal-Filling-OCS from [Hosseini et al., 2024].

Proof. Let $f(x)$ denote the number of agents who are matched at least $x \in [0, 1]$ (where $x = 1$ would imply the agent is saturated) under the divisible matching that corresponds to the probabilities from RANDOM. At each step of the online input stream, an item $o \in M$ arrives and the random algorithm produces a vector $(\tilde{x}_{a,o})_{a \in N}$ of probabilities for selecting each agent as follows: if an agent does not like the item, then $x_{a,o} = 1$, otherwise we set this value to be the probability of its selection by RANDOM. We then select an agent to be matched to the item with probability $\tilde{x}_{a,o}$.

By the end of the item arrivals, each agent is selected to match to an arriving item with probability

$$1 - \prod_{o \in M} (1 - \tilde{x}_{a,o}) \geq 1 - e^{-\sum_{o \in M} \tilde{x}_{a,o}}.$$

We henceforth denote the above probability bound by $p(\tilde{x}_a)$ for brevity. We can therefore bound

the expected value of the matching to a class i by:

$$\begin{aligned}\mathbb{E}\left[\, V_i(X) \,\right] &\geq \sum_{a \in N_i} p(\tilde{x}_a) \\ &= -\int_0^\infty p(\theta) df(\theta) \\ &= \int_0^\infty p'(\theta) f(\theta) d\theta\end{aligned}$$

where the first equality comes from the definition of $f(\theta)$ and the last from an integration by parts. As noted in [Hosseini et al., 2024], we have that for any divisible matching $\tilde{X}$ denoted by $\tilde{Y}_i$ for $i \in [k]$ such that $\sum_{i \in [k]} \tilde{Y}_{i,o} = 1$ for each $o \in M$:

$$\sum_{j \in [k]} V_i^*(\tilde{Y}_j) \leq k \cdot \left(\int_0^\theta f(z) dz + f(\theta) \right) \quad \forall \theta > 0.$$

Therefore, the prop_i value (which is a maximization over *divisible matchings*) can be upper bounded as

$$\text{prop}_i \leq \int_0^\theta f(z) dz + f(\theta)$$

for all $\theta > 0$. We can further multiply this bound by non-negative coefficients and integrate with respect to θ to obtain

$$\begin{aligned}\text{prop}_i \cdot \int_0^\infty c(\theta) d\theta &\leq \int_0^\infty c(\theta) \left(\int_0^\theta f(z) dz + f(\theta) \right) d\theta \\ &= \int_0^\infty c(\theta) \int_0^\theta f(z) dz d\theta + \int_0^\infty c(\theta) f(\theta) d\theta \\ &= \int_0^\infty \left(\int_z^\infty c(\theta) d\theta + c(z) \right) f(z) dz\end{aligned}$$

If we now choose the coefficients so that the relation $\int_z^\infty c(\theta)d\theta + c(z) = p'(z)$ holds for all $z > 0$ we obtain that

$$\text{prop}_i \cdot \int_0^\infty c(\theta)d\theta \leq \int_0^\infty p'(z)f(z)dz \leq \mathbb{E}\left[\, V_i(X)\,\right].$$

We lastly obtain the approximation factor by directly computing the integral

$$\begin{aligned}\int_0^\infty c(\theta)d\theta &= \int_0^\infty -e^\theta \int_\theta^\infty p''(y)e^{-y}dyd\theta \\ &= \int_0^\infty e^\theta \int_\theta^\infty e^{-2y}dyd\theta \\ &= \frac{1}{2}\int_0^\infty e^{-\theta}d\theta = \frac{1}{2}.\end{aligned}$$

Thus, we have the result. □

The synthesis of the above facts completes the proof of the theorem. □

5.7.2 CEF Upper Bounds

Indivisible Setting. We being by proving the CEF upper bound for indivisible matchings.

Proof of Theorem 5.1.3. Let τ denote the step in the input stream where no further items can be matched to the first class and note that $\tau \geq \frac{n}{2}$. Further note that, after step τ, all items will be matched to class 2. Let $n_1(t)$ be the random variable denoting the number of available agents in class 1 for the item arriving at time t and let $x(t)$ denote the number of items remaining in the input stream. We additionally let OPT_t denote the optimal matching agent in class 1 at time t. Further denote $\Delta n_1 = n_1(t) - n_1(t-1)$ and $\Delta x = x(t) - x(t-1)$. Naturally, $\Delta x = -1$ after every iteration, but for the $n_1(t)$ value we must consider three potential scenarios:

- $\Delta n_1 = 0$: this occurs when OPT_t is already saturated and the arriving item is matched to class 2.

- $\Delta n_1 = -2$: this occurs when OPT_t is unsaturated and the arriving item is not optimally matched to this agent.

- $\Delta n_1 = -1$: this occurs in all other events.

Using these facts, we obtain the following lemma.

Lemma 5.7.4. *In the setting of the proof of Theorem 5.1.3, the expected value of τ is $\frac{n}{e^2} + o(n)$.*

Before proving the lemma, we demonstrate how it is used in conjunction with the optimality of RANDOM for the given instance to complete the theorem's proof. Observe that

$$\begin{aligned}
\mathbb{E}\left[\, V_1(X) \,\right] &= \sum_t \Pr\left[\, o_t \text{ matched to class } 1 \,\right] \\
&= \sum_t \frac{1}{2} \Pr\left[\, n_1(t) \neq 0 \,\right] \\
&= \frac{1}{2} \sum_t \Pr\left[\, n_1(t) > 0 \,\right] \\
&= \frac{1}{2} \mathbb{E}\left[\, \tau \,\right]
\end{aligned}$$

where the second equality draws from the fact that, while there is an available matching in class 1, an item has a $\frac{1}{2}$ probability of being matched to that class. Lastly, combining with the result of Lemma 5.7.4 we have

$$\mathbb{E}\left[\, V_1(X) \,\right] = \frac{1}{2}\left(n - \frac{n}{e^2} - o(n)\right)$$

with the remaining items going to class 1. Comparing the two class matching sizes obtains the

desired bound of $\frac{1-1/e^2}{1+1/e^2} \approx 0.762$.

Lastly, by invoking the final key lemma below, we obtain the result.

Lemma 5.7.5. *The CEF performance of any non-wasteful online matching algorithm is upper bounded by the expected size of the matching produced by the* RANDOM *algorithm on the instance of instance* (I, π).

Thus, the competitive ratio proved above is the best achievable for the given instance and we are done. □

It remains to verify the two crucial lemmas leveraged in the proof above.

Proof of Lemma 5.7.4. We proceed computing the expected value of Δn_1 under two different conditions: OPT_t being saturated or not.[3]

First, assume OPT_t is saturated at some earlier iteration. Then, with probability $\frac{1}{2}$ the item arriving at time t is matched to class 2 and $\Delta n_1 = 0$, otherwise we must have that $\Delta n_1 = -1$. Therefore, we obtain

$$\mathbb{E}\left[\, \Delta n_1 | \text{OPT}_t \text{ saturated} \,\right] = \frac{1}{2}(-1) + \frac{1}{2}(0) = -\frac{1}{2}.$$

Next, we assume OPT_t is unsaturated. Again, with probability $\frac{1}{2}$ the arriving item is matched to class 2 and we decrease by -1. If, instead, the item is matched to class 1 then Δn_1 can be either -1 or -2 depending on how the item is matched. Since at time t there are $n_1(t)$ potential agents in class 1, the probability that the item is matched optimally is $\frac{1}{n_1(t)}$. Thus,

[3]Note that we are also implicitly assuming throughout this argument that $n_1(t) > 0$, i.e., at least one agent is still unsaturated in class 1.

we have

$$\begin{aligned}\mathbb{E}\left[\,\Delta n_1 | \text{OPT}_t \text{ unsaturated}\,\right] &= \frac{1}{2}(-1) + \frac{1}{2}\left(\frac{1}{n_1(t)}(-1) + \left(1 - \frac{1}{n_1(t)}\right)(-2)\right) \\ &= \frac{1}{2}\left(\frac{1}{n_1(t)} - 3\right).\end{aligned}$$

It remains to compute the probability that OPT_t is saturated. Note that by construction of our instance and the random algorithm, the probability that OPT_t is unsaturated by time t is exactly equal to the number of $n_1(t)$ sized subsets of $\{1, \ldots, x(t)\}$ which include the optimal agent [Karp et al., 1990]. Thus,

$$\Pr\left[\,\text{OPT}_t \text{ unsaturated}\,\right] = \frac{n_1(t)}{x(t)}.$$

We can now finally compute that

$$\mathbb{E}\left[\,\Delta n_1\,\right] = -\frac{1}{2} \cdot \left(1 - \frac{n_1(t)}{x(t)}\right) + \frac{1}{2} \cdot \frac{n_1(t)}{x(t)}\left(\frac{1}{n_1(t)} - 3\right).$$

Rearranging and simplifying we obtain that

$$\mathbb{E}\left[\,\Delta n_1\,\right] = -\frac{1}{2}\left(1 + \frac{2n_1(t) - 1}{x(t)}\right) \Rightarrow \frac{\mathbb{E}\left[\,\Delta n_1(t)\,\right]}{\Delta x(t)} = \frac{1}{2}\left(1 + \frac{2n_1(t) - 1}{x(t)}\right)$$

and by Kurtz' theorem [Kurtz, 1970], this is closely approximated by the following differential equation with probability tending to 1 as n tends to infinity:

$$\frac{dn_1}{dx} = \frac{1}{2}\left(1 + \frac{2n_1 - 1}{x}\right)$$

Solving with the initial condition that $n_1 = x = n$ and setting the resultant equation equal to 0, we obtain that the expected stopping time is $\tau = n(1 - \frac{1}{e^2}) - o(n)$. □

Proof of Lemma 5.7.5. We first note that any item in instance (I, π) allocated to class 2 can only be matched to one specific agent, so there is no decision to be made for items in this class. Moreover, for items that are allocated to class 1, the best possible matching is achieved in expectation by allocating completely at random to the potential agents, as proven in [Karp et al., 1990]. We therefore need only prove that RANDOM is optimal at the *class* level. We proceed to verify this by comparing with a general algorithm representative of the other possible class matchings.

For any arriving item, if only one class has a potential matching then by non-wastefulness we must give the item to that class. Therefore, assume item $o \in M$ arrives and can be matched to either class (i.e., both have a potential matching to a currently unsaturated agent). However, from the perspective of the algorithm, there is no way of distinguishing the two classes and any bias towards one can be exploited by the adversary by flipping the given problem instance. Thus, the best we can do is randomly pick one of the two classes to match the item and we have the result of the lemma. □

Divisible Setting. We now prove the bound of Theorem 5.4.2. Assume we have an algorithm that guarantees β-CEF.

Lemma 5.7.6. *To ensure a maximal number of agents in N_1 are saturated, it is optimal to distribute arriving items equally among the agents in this class.*

Proof of Lemma 5.7.6. For some i, consider the associated items $2i - 1$ and $2i$. We argue that it is optimal to distribute these two items equally within N_1.

By the adversarial construction of the adjacency in N_1, we know that one of the agents who likes these items does not like any of the following items. In the offline setting, it is straightforward to match the items to their associated agent and ensure a perfect matching. However, in the online setting, we do not know which of the current agents will be unavailable for future matching in the adversarial input stream. Therefore, in maximizing the USW, it is optimal to distribute this item equally within the first class. □

Now, let $\alpha = \frac{1-\beta}{1+\beta}$ and reciprocally define $\beta = \frac{1-\alpha}{1+\alpha}$. We proceed to demonstrate that any β-CEF algorithm must allocate arriving items in a strategic manner between the two classes of the given adversarial input to maintain this approximate guarantee.

Lemma 5.7.7. *Upon arrival of item o_t, if there exists $a \in N_1$ such that $\sum_{k<t} x_{a,o_k} < 1$ then any β-CEF algorithm must divide o_t among the two classes such that the first class receives $\frac{1+\alpha}{2}$ and the second receives $\frac{1-\alpha}{2}$.*

Intuitively, we demonstrate that the above ratio is optimal when working against an adversary who can easily flip the input instance of Figure 5.2b for the two classes if the algorithm biases its decision making too heavily. We then show that the β ratio is the best possible strategy against the possible adversarial input instances.

Proof of Lemma 5.7.7. Our algorithm must ensure that neither class is envious of the other (more than the aforementioned threshold, β). Otherwise, the adversary has an instance that one class is envious of the other one and contradicts the threshold. Here, we show that if it at any iteration the β-CEF guarantee holds, we can strategically divide the next item to maintain the necessary inequality.

For the given construction, we can always match items to N_2 which implies that $V_2^*(Y(X_1)) = V_1(X_1)$. Therefore, in achieving a β-CEF matching, we must match as much of each item to the first class as possible to achieve equitable treatment. In other words, we give the maximum possible proportion of each item to the first class in a way that it does not contradict the β-CEF constraint.

We will proceed by induction argument on the input stream. For the base case, assume towards contradiction that the first arriving item is not allocated according to the specified distribution, i.e. N_1 is not fully saturated and the item is not distributed with ratio $1+\alpha$ to $1-\alpha$ to the first and second classes respectively. Observe that we cannot give more than a $\frac{1+\alpha}{2}$ fraction of the arriving item to N_1 without violating the β-CEF guarantee.

For each of the subsequent items, we must match up to a $\frac{1+\alpha}{2}$ fraction of the item or less if all agents become saturated in N_1. In each instance, the remaining portion of the item is given to the second class. We proceed to prove that if the allocation was β-CEF after each item's division, then this property must persist. To this end, assume that the argument holds for $t-1$ for some $t \geq 2$.

First, we examine the second classes' perspective. We claim that matching the arriving item at iteration $t > 1$ according to the prescribed ratio ensures $V_2(X_2^t) \geq \beta \cdot V_1(X_1^t)$. We must have that this inequality was true prior to iteration t by the induction assumption and by matching item o_t according to the distribution between the two classes. Assume towards contradiction that we do not match according to the defined ratios at iteration t—by matching the arriving item with a portion higher than $\frac{1+\alpha}{2}$, an adversary can instead flip the input instance and force the algorithm to allocate a smaller portion of the item than is needed for the β-CEF guarantee. Thus, after allocating item o_t we cannot have given more than $\frac{1+\alpha}{2}$ to the first class, and thus retain

$V_2(X_2) \geq \beta \cdot V_1(X_1) = \beta \cdot V_2^*(Y(X_1))$, where the last equation follows from the fact that we can always match every item to N_2 that was matched to N_1.

We lastly claim that $V_1(X_1) \geq \beta \cdot V_1^*(Y(X_2))$. From the perspective of the first class, we must have that the β-CEF inequality holds after allocating the first item according to the defined ratio. Following the prior item's arrival, we demonstrated that the inequality held. From the construction, we further have that each item is connected to more agents compared to items arriving later. To be more precise, if item o_1 arrives before item o_2, then o_1 is connected to a set of agents which supersedes that of o_2. As a result, when we divide the arriving item, we have possibly decreased the portion of the next items (which were connected to fewer agents in class 1) and instead increased the portion of the current item (which is connected to a superset of the agents the next items are connected to). Therefore, if previously we had $V_1(X_1) \geq \beta \cdot V_2(X_2)$, then the inequality persists.

□

Combining the results of Lemma 5.7.6 and Lemma 5.7.7, we obtain conditions under which we maintain a β-CEF approximation. It remains to analyze the maximum such β value which satisfies these conditions.

Proof of Theorem 5.1.4. We begin by bounding the size of the matching to each class. First, we claim that N_2 will be saturated and its valuation will be n. Since we have $2n$ items and our algorithm is non-wasteful, we must match items in each step until the classes are satisfied. Further note that N_2 will *always* have an available matching by construction. Now, since N_1 can only match at most n items, we must have that $V_2(X_2) = n$.

We next bound the final step, i, after which no arriving items will be matched to N_1 as a

result of our equal distribution within the class (as proven in Lemma 5.7.6). By synthesis of the two above lemmas, we have that N_1 receives a $\frac{1+\alpha}{2} \cdot \frac{1}{n}$ portion of the first two items. Furthermore, the i-th pair of items will contribute $\frac{1+\alpha}{2} \cdot \frac{1}{n-i+1}$ to the size of N_1's matching.Therefore, it is sufficient to find the first value of i such that

$$\frac{1+\alpha}{2} \sum_{0 \le k < i} \frac{1}{n-k} = \frac{1+\alpha}{2} \left(H_n - H_{n-i}\right) \ge \frac{1}{2},$$

where H_n is the n-th Harmonic number. By the Euler–Maclaurin formula [Apostol, 1999], we have

$$H_n = \ln n + \gamma + \frac{1}{2n} - \epsilon_n, \text{ for } 0 \le \epsilon_n \le \frac{1}{8n^2}$$

where γ is the Euler-Mascheroni constant. We now define $\epsilon = \frac{1}{2n} - \epsilon_n - \left(\frac{1}{2(n-i)} - \epsilon_{n-i}\right)$ and rewrite the desired expression as

$$(1+\alpha)(H_n - H_{n-i}) = (1+\alpha)(\ln n - \ln(n-i) + \epsilon) \ge 1$$

This further implies

$$\ln n - \ln(n-i) \ge \frac{1}{1+\alpha} - \epsilon$$

which is equivalent to the following

$$\frac{i}{n} \le 1 - e^{-\frac{1}{1+\alpha}+\epsilon}. \tag{5.1}$$

Since i is the last item that is partially matched to N_1, Lemma 5.7.7 guarantees that this

class receives a $1+\alpha$ fraction of the item. Therefore, we must have that the matching to N_1 is

$$\beta \leq \frac{i(1+\alpha)}{n},$$

where the factor of n comes from the bound on N_2's matching and the inequality from the algorithm's approximation guarantee. Expanding on this ineqaulity using Equation (5.1) implies

$$\begin{aligned}
\beta &\leq \frac{i(1+\alpha)}{n} \\
&\leq (1+\alpha)\left(1 - e^{-\frac{1}{1+\alpha}+\epsilon}\right) \\
&= \frac{2}{1+\beta}\left(1 - e^{-\frac{(1+\beta)}{2}+\epsilon}\right) \\
&= \left(1 - e^{\frac{-(1+\beta)}{2}} \cdot e^{\epsilon}\right)\frac{2}{1+\beta},
\end{aligned}$$

It remains show the limiting behavior of this bound on our approximation ratio as n increases.

Claim 5.7.8. $\lim_{n\to\infty} \epsilon = 0$

Proof. By definition of ϵ and the bound of $\epsilon_n < \frac{1}{8n^2}$, we can compute

$$\begin{aligned}
|\epsilon| &= \left|\frac{1}{2n} + \frac{1}{2(n-i)} + (\epsilon_{n-i} - \epsilon_n)\right| \\
&\leq \frac{1}{2n} + \frac{1}{2(n-i)} + \frac{1}{8(n-i)^2}
\end{aligned}$$

Now by invoking Equation (5.1) we can expand the definition of ϵ as

$$|\epsilon| \leq \frac{1}{2n} + \frac{1}{2n} \cdot e^{\frac{1}{1+\alpha}-\epsilon} + \frac{1}{8}\left(\frac{1}{n} \cdot e^{\frac{1}{1+\alpha}-\epsilon}\right)^2$$

We lastly compute the limit:

$$\begin{aligned}
\lim_{n\to\infty} |\epsilon| &\leq \lim_{n\to\infty} \left(\frac{1}{2n} + \frac{1}{2n} \cdot e^{\frac{1}{1+\alpha}-\epsilon} + \frac{1}{8}\left(\frac{1}{n}e^{\frac{1}{1+\alpha}-\epsilon}\right)^2\right) \\
&\leq \lim_{n\to\infty} \left(\frac{1}{2n} + \frac{1}{2n} \cdot e^{\frac{1}{1+\alpha}-2} + \frac{1}{8}\left(\frac{1}{n}e^{\frac{1}{1+\alpha}-2}\right)^2\right) \\
&= 0
\end{aligned}$$

completing the claim. □

Using the result of Claim 5.7.8, we have that $\lim_{n\to\infty} e^{\epsilon} = 1$ and thus by taking $n \to \infty$ we have

$$\beta \leq \left(1 - e^{\frac{-(1+\beta)}{2}}\right) \frac{2}{1+\beta}$$

By direct computation, we obtain that $\beta \leq 0.677$. Therefore, we cannot achieve β-CEF for $\beta > 0.677$. □

5.7.3 Price of Fairness

Proof of Theorem 5.5.1. Suppose an algorithm $\mathcal{A}$ is α-CEF for some $\alpha \in (0, 1)$. Let p and q be coprime integers such that $|\alpha - p/q| < \epsilon$ for some arbitrarily small $\epsilon > 0$. Assume further that we have $k-1$ classes comprised of q agents, as well as a k-th class comprised of $q(k-1)$ agents. An adversary constructs an input stream wherein the items arrive in two phases. In the first phase,

$p(k-1)+q$ items arrive, each of which has edges to *every* agent in the graph. The second phase consists of $k-1$ groups arriving sequentially, where the i-th such group is comprised of q items with edges to all the agents in class i. Let $c_i(t)$ be a random variable that indicates the number of items allocated to class i at the end of round t and let $\tau = p(k-1)+q$. We first prove the following claim for the given instance.

Claim 5.7.9. *For any non-wasteful α-CEF algorithm and $\tau = p(k-1)+q$, we must have that*

$$\mathbb{E}\left[\sum_{i=1}^{k-1} c_i(\tau)\right] \geq p(k-1).$$

Proof. Assume towards contradiction that $\mathbb{E}\left[\sum_{i=1}^{k-1} c_i(\tau)\right] < p(k-1)$. Since the algorithm is assumed to be non-wasteful, this implies that the k-th class must have $\mathbb{E}[\, c_k(\tau)\,] \geq q$ to account for all the items arrived thus far. By the pigeonhole principle, our assumption further implies that there exists some $i \in [k-1]$ such $\mathbb{E}[\, c_i(\tau)\,] < p$. Thus, we must have that

$$V_i(X) = \mathbb{E}[\, c_i(\tau)\,] < p < q \leq \mathbb{E}[\, c_k(\tau)\,] = V_i^*(Y_k(X))$$

contradicting our assumption on the α-CEF guarantee. □

We now proceed to analyze the class matching size in the second phase of item arrivals. Of the q arriving items specific to some class i, exactly $c_i(\tau)$ must remain unmatched since all feasible agents will already be saturated. This implies that the utilitarian social welfare at the end

of the second phase is

$$\text{USW}(X) = \left(\sum_{i=1}^{k-1} c_i(\tau) + c_k(\tau)\right) + \sum_{i=1}^{k-1}(q - c_i(\tau)) = \sum_{i=1}^{k} c_i(\tau) + \sum_{i=1}^{k-1} q - \sum_{i=1}^{k-1} c_i(\tau)$$

Now, using the fact that $p(k-1)+q$ items arrive in the first phrase where all can be matched to any agents, we further have that

$$\text{USW}(X) = p(k-1) + q + \sum_{i=1}^{k-1} q - \sum_{i=1}^{k-1} c_i(\tau) = p(k-1) + qk - \sum_{i=1}^{k-1} c_i(\tau)$$

where the remaining equalities are mere algebra manipulation on the summations. Now, as a result of Claim 5.7.9 we have that in expectation:

$$\mathbb{E}\left[\,\text{USW}(X)\,\right] = \mathbb{E}\left[\,p(k-1) + qk - \sum_{i=1}^{k-1} c_i(\tau)\,\right] \leq qk.$$

Lastly, observe that the optimal offline solution for this adversarial input instance would instead allocate all items in the first phase to class k, and the remaining q items specific to each class to their corresponding class, giving a USW value of $p(k-1)+q+q(k-1)$. Thus, the competitive ratio for the USW objective is given by

$$\frac{qk}{qk + p(k-1)} = \frac{1}{1 + \alpha(k-1)/k}$$

which tends towards a lower bound of $\frac{1}{1+\alpha}$ as k increases. Thus, we have the result of the theorem.

□

5.7.4 Nash Social Welfare

Proof. Consider a matching X that maximizes the CNSW objective. We first note that any wasteful matching can necessarily increase the CNSW objective by matching any wasted item, so we have non-wastefulness must hold.

Now, towards contradiction, suppose that X does not satisfy CEF1. By the definition of CEF1, this implies that there exists two classes $i, j \in [k]$ such that

$$V_i(X) < V_i^*(Y_j(X) \setminus \{o\}),$$

for some $o \in Y_j(X)$. Let $V_i(X) = |X_i| = s$. By the assumption above, we must have

$$V_i^*(Y_j(X) \setminus \{o\}) > V_i(X) = |X_i| = s,$$

and thus $|Y_j(X) \setminus \{o\}| \geq s + 1$ for some $o \in Y_j(X)$ since s is an integer. Combining this result with the non-wastefulness of a CNSW maximizing matching, we further have that $V_j(X) = |X_j| = |Y_j(X)| \geq s + 2 > V_i(X) + 1$. We now verify the following claim that will be crucial in proving the final CEF1 result.

Claim 5.7.10. *There exists an item $o' \in Y_j(X)$ such that*

$$V_i(X_i \cup \{o'\}) > V_i(X_i)$$

Before proving this claim, we show how it yields the CEF1 guarantee: starting from matching X, consider a modified allocation X' that moves the item $o' \in Y_j(X)$ from Claim 5.7.10 to

X_i. We compute the CNSW of this modified matching as

$$\begin{aligned}\text{CNSW}(X') &= \left((V_i(X)+1)\cdot(V_j(X)-1)\prod_{[k]\ni s\neq i,j} V_s(X)\right)^{1/k} \\ &\geq \left(V_i(X)\cdot V_j(X)\prod_{[k]\ni s\neq i,j} V_s(X)\right)^{1/k} \\ &= \text{CNSW}(X)\end{aligned}$$

where the inequality follows from the contradictory assumption. Naturally, this contradicts the maximality of $\text{CNSW}(X)$, thus we must have that X is a CEF1 matching. □

We conclude the theorem's proof by verifying Claim 5.7.10.

Proof of Claim 5.7.10. Fix an item $o \in Y_j(X)$ and let S_i denote the set of agents in class i who are matched under X. We similarly define S_i^* be the set of saturated agents in class i under the optimistic matching of items in $Y_j(X) \setminus \{o\}$. Since $V_i^*(Y_j(X) \setminus \{o\}) > Y_i(X)$, we have $|S_i^* \setminus S_i| \geq 1$.

Now, pick any agent $a' \in S_i^* \setminus S_i$, and let o' denote the item that is matched to a' under the optimistic matching on $Y_j(X) \setminus \{o\}$. By the selection of a' and o', we must have that o' can be matched to agent a' without any conflicts. Therefore, $V_i(Y_i(X) \cup \{o'\}) > V_i(X)$. □

Proof of Theorem 5.6.3. Consider two classes N_1 and N_2, each of which consists of four agents, denote these agents by a_1, a_2, a_3, a_4 and b_1, b_2, b_3, b_4 respectively. Fix an adversarial sequence of six items indexed by $1, 2, \ldots, 6$. Items $1, 2, 3$ and 4 have edges with all the agents in class N_2, and item 1 further has an edge with agent a_1. Items 5 and 6 have edges with agents a_3 and a_4. Consider a matching X that matches items $1, 2, 3, 4$ to class N_2 and $5, 6$ to class N_1 (See Figure 5.3 for a

depiction). Due to the construction of edges, this must be a non-wasteful matching and the NSW is computed to be $\sqrt{2 \cdot 4}$. Note further that allocating items $1, 2, 3, 4$ to class N_1 only induces a matching to class N_2 of optimistic value $V_2(Y_1^*(X) = 2$, and thus X is CEF. On the flip side, if we consider an allocation X' that allocates items $2, 3, 4$ to class N_2 and $1, 5, 6$ to class N_1, this also constitutes a non-wasteful matching and the NSW is $\sqrt{3 \cdot 3} > \sqrt{2 \cdot 4}$. Therefore, a non-wasteful CEF matching does not necessarily maximize the CNSW objective. □

5.8 Conclusions, Limitations & Future Work

Our work closes the long-standing open conjecture on whether a non-wasteful randomized algorithm can achieve non-trivial fairness guarantees in the context of online matching problems. By conducting a detailed and non-trivial analysis of a natural randomized matching procedure, we have successfully developed an algorithm that not only complements our previously established 0.76-CEF upper bound construction but also aligns with the known upper bounds for the CPROP and USW efficiency guarantees.

Moreover, we demonstrate that the algorithmic guarantee of [Hosseini et al., 2024] for deterministic and divisible matchings is almost tight, with a novel input construction that exhibits a 0.67-CEF upper bound. We lastly define "price of fairness" for the online matching problem and present an interesting impossibility result on the trade-off between fair and optimal solutions. Namely, we demonstrate that any approximation to the CEF objective follows an inverse proportionality relationship to the possible USW approximation any algorithm can obtain. This demonstrates that we must allow some degradation in solution quality to ensure equitable treatment of classes.

Future work should address the natural gaps between the upper and lower bounds discussed above. Perhaps most importantly, can a randomized algorithm achieve a USW approximation better than $\frac{1}{2}$ while maintaining the given fairness guarantees? Is the price of fairness result tight, ie. does an algorithm exist that ensures α-CEF while simultaneously guaranteeing $\frac{1}{1+\alpha}$-USW? We believe that some of these answers may result from a careful extension on the RANKING algorithm that will naturally rely on some priority ordering over both classes and agents.

Part III

Fair Hierarchical Clustering

Chapter 6: Generalized Reductions: Making any Hierarchical Clustering Fair and Balanced

6.1 Introduction

Fair machine learning, and namely clustering, has seen a recent surge as researchers recognize its practical importance. In spite of the clear and serious impact the lack of fairness in existing intelligent systems has on society [Angwin et al., 2016a, Rieke and Bogen, 2018, Ledford, 2019, Sweeney, 2013], and despite significant progress towards fair flat (not hierarchical) clustering [Ahmadian et al., 2020b, Backurs et al., 2019, Bera et al., 2019, Brubach et al., 2020, Chakrabarti et al., 2022, Chen et al., 2019, Chierichetti et al., 2017, Esmaeili et al., 2021, Esmaeili et al., 2020, Kleindessner et al., 2019a, Rösner and Schmidt, 2018], fairness in hierarchical clustering has only received some recent attention [Ahmadian et al., 2020a, Chhabra and Mohapatra, 2022]. Thus, we are some of the first to study this problem.

Hierarchical clustering (Figure 6.1) is the well-known extension to clustering, where we create a hierarchy of subclusters contained within superclusters. This structure forms a tree (a dendrogram), where leaves represent the input data. An internal node v corresponds to the cluster of all the leaves of the subtree rooted at v. The root is the cluster containing all data points.

Hierarchical clusterings more completely illustrate data relationships than flat clusterings.

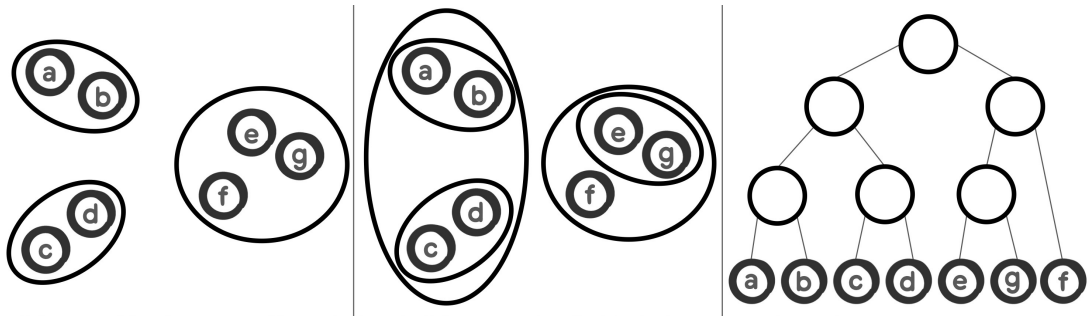


Figure 6.1: On the left is a 3-clustering, in the center is a hierarchical clustering, and on the right is its dendrogram.

For instance, they are commonly used in phylogenetics to depict the entire evolutionary history of species, whereas a clustering would only depict species similarities. It also has a myriad of other uses in machine learning applications such as search [Cai et al., 2004, Ferragina and Gulli, 2005, Kou and Lou, 2012], social network analysis [Leskovec et al., 2014, Mann et al., 2008], and image recognition [Arifin and Asano, 2006, Lin et al., 2018, Pan et al., 2016]. On top of this, hierarchical clusterings can also be used to solve flat clustering when the number of clusters is not given. To do this, we extract clusterings at different resolutions in the hierarchy that all "agree" (if two points are together in a cluster, then they will also be together in any larger cluster) and select the one that best fits the application.

Hierarchical clusterings can be evaluated using a number of metrics. Perhaps most notably, [Dasgupta, 2016] introduced cost (Definition 6.2.2), an intuitive and explainable metric which exhibits numerous desirable properties and has become quite popular and well respected [Charikar and Chatziafratis, 2017, Charikar et al., 2019, Chatziafratis et al., 2018, Cohen-Addad et al., 2017, Roy and Pokutta, 2016]. Unfortunately, this objective is difficult to approximate, where the best existing solutions require semi-definite programs [Dasgupta, 2016, Charikar and Chatziafratis, 2017], and it is not efficiently $O(1)$-approximable by the Small-Set Expan-

sion Hypothesis [Charikar and Chatziafratis, 2017]. The revenue [Moseley and Wang, 2017] and value [Cohen-Addad et al., 2018] metrics, both derived from cost, exhibit $O(1)$-approximability, but are not as explainable or appreciated.

Only two papers have explored fair hierarchical clustering [Ahmadian et al., 2020a,Chhabra and Mohapatra, 2022]. Both extend fairness constraints from fair clustering literature that trace back to [Chierichetti et al., 2017]'s *disparate impact*. More formally, consider a graph $G = (V, E, w)$, where each point has a color representing some protected class (e.g., gender, race, etc.). On two colors, red and blue, they deem a clustering to be *fair* if the ratio between red and blue points in each cluster is equal to that in the input data. This ensures that the impact of a cluster on a protected class is proportionate to the class size. The constraint has been further generalized [Ahmadian et al., 2019, Bercea et al., 2019]: given a dataset with λ colors and constraint vectors $\vec{\alpha}, \vec{\beta} \in (0, 1)^\lambda$, a clustering is fair if for all $\ell \in [\lambda]$ and every cluster C, $\alpha_\ell |C| \leq \nu(C, \ell) \leq \beta_\ell |C|$, were $\nu(C, \ell)$ is the number of points in C of color ℓ. Naturally, then, a hierarchical clustering is fair if every non-singleton cluster in the hierarchy satisfies this constraint (with nuances, see Section 6.2.2), as in [Ahmadian et al., 2020a].

This chapter explores broad guarantees, namely cost approximations, for fair hierarchical clustering. The only previous algorithm is quite complicated and yields a $O(n^{5/6} \log^{5/4} n)$ fair approximation for cost, only slightly beating the trivial $O(n)$-approximation [Ahmadian et al., 2020a]. Moreover, the algorithm assumes the input contains only two, equally represented, colors—an exceedingly strong assumption in modern datasets. This reflects the inherent difficulty of finding solutions that are low-cost as opposed to high-revenue or high-value, both of which exhibit fair $O(1)$-approximations [Ahmadian et al., 2020a]). Our algorithms improve the prior work in several ways:

	Property	Approximation	Fairness	Colors	Color ratios	Explainable?
Prior	Determ. fair	$O(n^{5/6}\text{polylog } n)$	Perfect	2	50/50 only	No
This chapter	Determ. fair	$O(n^{\delta}\text{polylog } n)$	Approx.	$O(1)$	$O(1)$	Yes
	Stoch. fair	$O(\log^{3/2} n)$	Approx.	$O(1)$	$O(1)$	Yes
	ϵ-rel. balance	$O(\sqrt{\log n}/\epsilon)$	N/A	N/A	N/A	Yes
	$\frac{1}{6}$-rel. balance	$O(\sqrt{\log n})$	N/A	N/A	N/A	Yes

Table 6.1: Results of Chapter 7 as compared to previous work. Note $\delta \in (0, 1/6)$ is parameterizable, trading approximation factor for fairness. Our algorithms are explainable in that the alterations made to the hierarchy are clear and well-defined.

1. We achieve a near-exponential improvement in approximation factor ($\text{poly}(n)$ to near-$\text{polylog}(n)$).

2. Our algorithm works on $O(1)$ instead of only 2 colors.

3. Our work handles different representational proportions across colors in the initial dataset.

4. We simultaneously guarantee fairness and relative cluster balance.

5. Our methods, which modify a given (unfair) hierarchy, have measurable, explainable, and limited impacts on the structure of the input hierarchy.

6.1.1 Our Contributions

This chapter proposes new algorithms for fair and balanced hierarchical clustering. A summary of our results can be found in Table 6.1.

A key component of our approach is the introduction of four simple hierarchy tree operators, each with a well-defined and measurable impact. We demonstrate how these operators can be systematically applied to an initially unbalanced or unfair hierarchy to transform it into a fairer and more balanced structure while preserving its overall form. This process not only

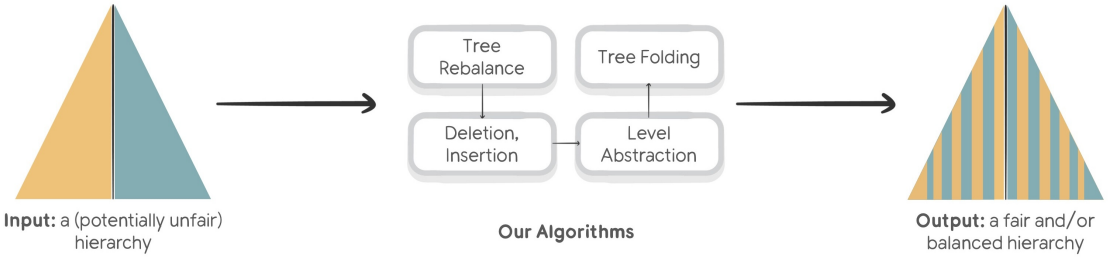


Figure 6.2: Our algorithms take a potentially unfair hierarchical clustering, apply our tree operators, and yield fair and/or balanced hierarchies.

highlights the functionality of our algorithms but also makes explicit the modifications applied to the hierarchy. Each of our four proposed algorithms begins with a given γ-approximate (unfair) hierarchical clustering algorithm, such as [Dasgupta, 2016]'s $O(\sqrt{\log n})$-approximation, and incrementally applies one of these operators to refine the clustering. While each algorithm builds on the previous ones, introducing additional improvements, they also stand independently as distinct contributions.

Our first algorithm produces a $\frac{1}{6}$-relatively balanced hierarchy that $\frac{3}{2}\gamma$-approximates cost (see Theorem 6.4.1).[1] Here, ϵ-relative balance means that at each split in the hierarchy, a cluster splits in half within a proportional error of at most $1+\epsilon$ (see Definition 6.3.1). Starting at the root, this algorithm recursively applies our *tree rebalance* (see Definition 6.3.4) operator. This restructures the tree by moving some subtree up to become a child of the root. This process preserves much of the hierarchy's structure while achieving relative balance.

Our next result refines this to achieve ϵ-relative balance for any $\epsilon \in \left(0, \frac{1}{6}\right)$ that $\frac{9}{2\epsilon}\gamma$-approximates cost (see Theorem 6.4.5). This can get arbitrarily close to creating a perfectly balanced hierarchy. To achieve this, we simply run our first algorithm and then apply a limited number of *subtree deletion and insertion operators* (see Definition 6.3.5). This operator selects a

[1]Repeated sparsest cuts achieves this with similar cost. Our algorithms, though, can be used explainably on top of existing unfair algorithms and may perform better as unfair research progresses.

subtree, removes it, and reinserts elsewhere. It again preserves much of T's structure.

Third, we propose an algorithm for stochastically fair hierarchical clustering (see Definition 6.2.4). Under certain stochastic parameterizations and arbitrarily many colors, the algorithm achieves *stochastic* fairness and $O(\gamma \log n)$-approximates cost (see Theorem 6.4.9). This is quite impressive, as the best previous fair approximation (albeit, for deterministic colors) was $poly(n)$ [Ahmadian et al., 2020a]. To achieve this novel result, we first find an $O(1/\log n)$-relatively balanced hierarchy and then apply our level abstraction operator once to the bottom layers of the hierarchy. This operator removes selected layers, setting much lower descendants of a vertex as direct children. While this removes details in the hierarchy, the remaining structure still agrees with the original tree. This simple operation guarantees fairness under stochastic color assignment.

Our main result leverages all of the above to find an approximately fair hierarchical clustering that $O(n^{\delta}\mathrm{polylog}\ n)$-approximates cost (see Theorem 6.4.14), where $\delta \in (0, 1)$ is a given constant. This is a near-exponential improvement over previous work which only achieves a $O(n^{5/6}\mathrm{polylog}\ n)$. On top of that, our algorithm works on many colors, with many different color ratios, and achieves a simultaneously balanced hierarchy in an explainable manner. The algorithm, FairHC (Algorithm 10), builds on top of our stochastic algorithm (parameterized slightly differently, see Section 6.4.4) before applying a new operator: tree folding. Tree folding maps isomorphic trees on top of each other. In hierarchical clustering, this means taking two subtrees, mapping clusters in one tree to the other, and then merging clusters according to the mapping. Matching up clusters with different proportions of colors helps balance out the color ratios across clusters, which gives us our fairness result.

Corollary 6.1.1. *Given a γ-approximation for* cost *and a $\frac{1}{c \log n}$-relatively balanced hierarchy tree T over $\nu(V, \ell) = c_\ell n = O(n)$ vertices for each color $\ell \in \lambda$ with $h = n^\delta$ for any constants c, δ, there is an algorithm that yields a $O(n^\delta \gamma \log^2 n)$ fair approximation for* cost*, provided that*

$$\beta_\ell \geq x_\ell c_\ell \text{ and } \alpha_\ell \leq (1 - x_\ell) c_\ell$$

for each $\ell \in [\lambda]$, where $x_R, x_B = O(1)$ and can be parameterized to be arbitrarily close to 1. This algorithm runs in $O(n^2 \log n)$ time.

6.2 Preliminaries

Our input is a complete weighted graph $G = (V, E, w)$ where $w : E \to \mathbb{R}^+$ is a similarity measure. A hierarchical clustering can be defined as a hierarchy tree T, where its leaves are $\text{leaves}(T) = V$, and internal nodes represent the merging of vertices into clusters and clusters into superclusters.

6.2.1 Optimization Problem

We use [Dasgupta, 2016]'s cost function as an optimization metric. For simplicity, let $n_T(e)$ denote the size of smallest cluster in T containing both endpoints of e. In other words, for $e = (u, v)$, $n_T(e) = |\text{leaves}(T[u \wedge v])|$, where $u \wedge v$ is the lowest common ancestor (LCA) of u and v in T and $T[u]$ is the subtree rooted at u for some vertex u. We further denote $n_T(u) = |\text{leaves}(T[u])|$ for internal (non-leaf) node, u. Let $\text{root}(T)$ be the root of T, and $\text{left}_T(u)$ and $\text{right}_T(u)$ access left and right children respectively.

Using this notation, can define the cost contribution of an edge to a hierarchy:

Definition 6.2.1. The **cost** of $e \in E$ in a graph $G = (V, E, w)$ in a hierarchy T is

$$\text{cost}_T(e) = w(e) \cdot n_T(e).$$

We thus strive to minimize the sum of costs across all edges:

Definition 6.2.2 ([Dasgupta, 2016])**.** The **cost** of a hierarchy T on graph $G = (V, E, w)$ is:

$$\text{cost}(T) = \sum_{e \in E} \text{cost}_T(e)$$

In the initial formulation, [Dasgupta, 2016] showed that any unfair tree optimizing for cost can be assumed to be binary. Note, however, that we must consider non-binary trees when we incorporate fairness, as it may not allow binary splits at its lowest levels.

6.2.2 Fairness and Stochastic Fairness

We consider the fairness constraints based off those introduced in [Chierichetti et al., 2017] and extended by [Bercea et al., 2019]. On a graph G with colored vertices, define $\nu(C, \ell)$ to be the number of λ-colored points in cluster C. We can then characterize a hierarchy as (demographically) with respect to these colorings.

Definition 6.2.3. Consider a graph $G = (V, E, w)$ with vertices colored one of $\lambda \in \mathbb{N}$ colors, and two vectors of parameters $\alpha, \beta \in (0, 1)^\lambda$ with $\alpha_\ell \leq \beta_\ell$ for all $\ell \in [\lambda]$. A hierarchy T on G is **fair** if, for any non-singleton cluster C in T, and for every $\ell \in [\lambda]$, $\alpha_\ell |C| \leq \nu(C, \ell) \leq \beta_\ell |C|$. Additionally, any cluster with a leaf child has only leaf children.

Notice that the final constraint regarding leaf-children simply enforces that a hierarchy must

have some "baseline" fair clustering (e.g., a fairlet decomposition [Chierichetti et al., 2017]). Consider a tree that is just a stick with individual leaf children at each depth. While internal nodes may represent fair clusters, you cannot extract any non-trivial fair flat clustering from this, since it must contain a singleton, which is unfair. Our additional constraint prevents this triviality.

In the stochastic problem, points are assigned colors at random. Thus, we must instead guarantee that all clusters are fair with high probability (i.e., at least $1 - 1/\text{polylog}(n)$).

Definition 6.2.4. Consider the same context as Definition 6.2.3, with an additional input function $p_\ell : v \to (0, 1)$ denoting the probability v has color ℓ such that $\sum_{\ell=1}^{\lambda} p_\ell(v) = 1$, and each vertex has exactly one color. An algorithm is **stochastically fair** if, with high probability, it outputs a fair hierarchy.

6.3 Tree Properties and Operators

While the core of this chapter is on yielding fair hierarchies from an unfair input, we further seek to construct *balanced* solutions. Balanced trees have numerous practical uses and, in this chapter, we show that they can further be used to guarantee a fair solution.

Definition 6.3.1. A hierarchy T is **ϵ-relatively balanced** if, for every pair of clusters C and C' that share a parent cluster C_p such that $|C_p| \geq 1/(2\epsilon)$ in T, then

$$(1/2 - \epsilon)|C_p| \leq |C|, |C'| \leq (1/2 + \epsilon)|C_p|.$$

Notice that we only care about splitting clusters C_p with size satisfying $|C_p| \geq 1/(2\epsilon)$. This is because, on smaller clusters, it may be impossible to divide them with relative balance. For

instance, if $|C_p| = 3$, we know we can only split it into a 1-sized and 2-sized cluster, yielding a minimum relative balance of $1/6$. For smaller ϵ, we require larger cluster sizes to make this possible.

We frequently refer to the "separation" of edges in our proposed operators. This occurs when a point is first added to a cluster that contains both of its endpoints by an operator. The removal of points is not a concern in our analysis.

Definition 6.3.2. An edge $e = (u, v)$ is (or its endpoints are) **separated** by an operator if it changes hierarchy T to T' such that $\text{cluster}_T(u \wedge v) \not\supseteq \text{cluster}_{T'}(u \wedge v)$.

Almost definitionally, if an edge is not separated by an operator, then the cluster size of its lowest common ancestor does not increase. Thus, its cost contribution does not increase. As we will see in Section 6.6, this observation is crucial to analyzing the degradation of solution optimality as we move from a (potentially) unfair hierarchy to a fair one.

6.3.1 Tree Operators

Our work employs various tree operations to modify and combine trees (Figure 6.3), providing a clear illustration of how our algorithms alter the input structure. We quantify the number of invocations of each type of operator in our algorithms and evaluate their impact on the hierarchy using a metric introduced in this work. Notably, each proposed algorithm transforms an input tree T into an output tree T' solely by applying our four tree operators: *tree rebalancing, subtree deletion and insertion, level abstraction, and subtree folding.*

Each of these four operators has an associated operation cost that measures the proportional increase in cost for each edge separated by the operation. We present lemmas that bound the

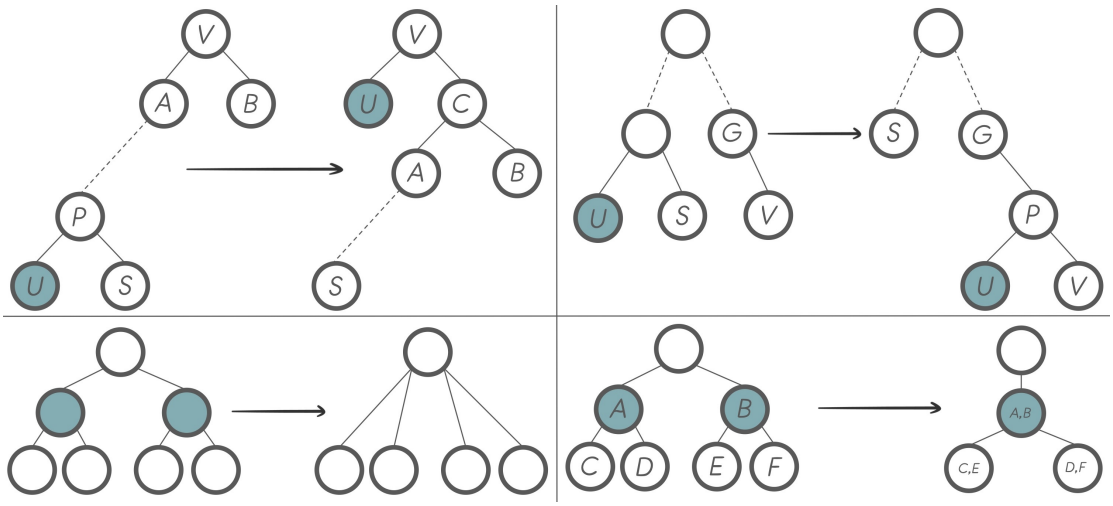


Figure 6.3: We depict our tree operators: tree rebalance (top left), subtree deletion and insertion (top right), level abstraction (bottom left), and tree folding (bottom right).

"operation cost" of each operator in Section 6.6.

Definition 6.3.3. Let T be a tree that undergoes a transformation via some tree operation, resulting in a new tree T'. The **operation cost** is an upper bound Δ such that for any edge e that is separated by the operation, the cost in the modified tree satisfies:

$$\text{cost}_{T'}(e) \leq \Delta \cdot \text{cost}_{T}(e).$$

The first operation we leverage is the "tree rebalance" which rotates a descendant of the root to instead be a direct child (see the top left panel of Figure 6.3 for an illustration). A clever, recursive, use of the tree rebalance operator allows us to find a relatively balanced tree (Theorem 6.4.1)

Definition 6.3.4. Consider a binary tree T with internal node v. Denote v's children by a and b, and let $u \in T[a]$ (without loss of generality). A **tree rebalance** of u at v ($\text{tree\_rebalance}(u, v)$) places a new node, c, between v and sibling nodes a and b. Subtree $T[u]$ is then removed from

$T[a]$ and is placed as v's other child.

Carefully applying the tree rebalance operator will only yield $1/6$-relatively balanced trees. Interestingly, [Dasgupta, 2016]'s sparsest cut algorithm, one of the current best cost approximations, achieves the same guarantee via a comparable analysis. To further refine this, we use subtree "insertions" and "deletions" (Figure 6.3, top right).

Definition 6.3.5. Consider a binary tree T with internal nodes u and v which are not ancestors of each other. Denote u's sibling by s, and v's parent by g. A **subtree deletion** at u removes $T[u]$ from T and contracts s into its parent. A **subtree insertion** of $T[u]$ at v inserts a new parent p between v and g, and adds u as a second child of p. The operator $\mathrm{del\_ins}(u, v)$ deletes u and inserts $T[u]$ at v.

In yielding a fair hierarchy, we will further need to abstract away (Figure 6.3, bottom left) certain levels of the hierarchy to simplify it. This involves taking vertices at depth d_2 and iteratively merging them into their parents until they reach depth d_1. In other words, ignore the tree structure between two levels of the hierarchy.

Definition 6.3.6. Consider a binary tree T with two parameters d_1 and d_2 such that $d_1 < d_2 < \mathrm{height}(T)$. **Level abstraction** between levels $d_1 + 1$ and d_2 ($\mathrm{abstract}(d_1, d_2)$) involves taking all internal nodes between depths $d_1 + 1$ and d_2 in T and contracting them into their parents.

Lastly, we use tree folding (Figure 6.3, bottom right) to map the topologies of multiple isomorphic trees (ignoring leaves) together.

Definition 6.3.7. Consider a set of subtrees $T_1, \ldots, T_k$ of T such that all trees T_i without their leaves have the same topology, and all $\mathrm{root}(T_i)$ have the same parent p in T. This means that for

each $i \in [k]$, there is a tree isomorphism $\phi_i : I_i \to I_k$ where I_i and I_k are the internal nodes of the corresponding trees. A **tree folding** of trees $T_1, \ldots, T_k$ ($\mathrm{fold}(T_1, \ldots, T_k)$) modifies T such that all $T_1, \ldots, T_k$ are replaced by a single tree T_f whose $\mathrm{root}(T)$ is made a child of p and T_f has the same internal topology as I_k such that for any leaf ℓ of any tree T_i with parent p_i in T_i, we set its parent to $\phi_i(p_i)$.

6.4 Fair and Balanced Reductions

We now present our main algorithms, which sequentially build on top of each other, adding new operators in a limited, measurable capacity to achieve new results.

6.4.1 Relatively Rebalanced Trees

Our first algorithm guarantees $1/6$-relative balance, modifying the input dendrogram using only a series of limited tree rebalances (Definition 6.3.4) We can show that this only incurs a small constant factor proportionate increase in cost.

Theorem 6.4.1. *Given a γ-approximation for cost, we can construct a $\frac{3}{2}\gamma$-approximation for cost which guarantees $\frac{1}{6}$-relative balance. The procedure modifies the tree by applying tree rebalance operators of operation cost $\frac{3}{2}$, and separates every edge by at most one such operator.*

Algorithm 8 RebalanceTree

Input: A hierarchy tree T of size n, with smaller cluster always on the left.
Output: A $\frac{1}{6}$-rebalanced T-tree.

$r, v = \text{root}(T)$
$A = \text{leaves}(\text{right}_T(v))$
while $|A| \geq \frac{2}{3}n$ **do**
    $v \leftarrow \text{right}_T(v)$
    $A \leftarrow \text{leaves}(\text{right}_T(v))$
end while
$T \leftarrow T.\text{tree\_rebalance}(v, r)$
Let $T'_l = \text{RebalanceTree}(T[\text{left}_T(r)])$
Let $T'_r = \text{RebalanceTree}(T[\text{right}_T(r)])$
Return T' with root r with $\text{left}(r) = \text{root}(T'_l)$ and $\text{right}(r) = \text{root}(T'_r)$

This idea was first introduced by [Dasgupta, 2016] as an analytical tool for their algorithm. However, we use it more explicitly here to take any given hierarchy and rearrange it to be balanced. We begin with a given tree T and construct a hierarchical partitioning as follows. First, draw T from top to bottom, ensuring that at each split, the smaller cluster is placed on the left. Let A_1 and B_1 denote the two clusters in the initial split. Proceed by recursively refining the left-most cluster, splitting A_1 into A_2 and B_2, and continuing this process iteratively. We terminate this sequence upon encountering the first cluster B_k such that $|B_k| \geq n/3$. This defines our first major partition: we divide the vertex set V into two subsets, A_k and $B = \bigcup_{i=1}^{k} B_i$. Finally, we apply this procedure recursively on both A_k and B to construct the full hierarchical decomposition. The search halting condition on this procedure then enforces that this yields a $\frac{1}{6}$-relatively balanced tree.

Lemma 6.4.2. *Algorithm 8 produces a $\frac{1}{6}$-relatively balanced tree.*

In the execution of RebalanceTree, we further show that once an edge is separated, it will never be separated again.

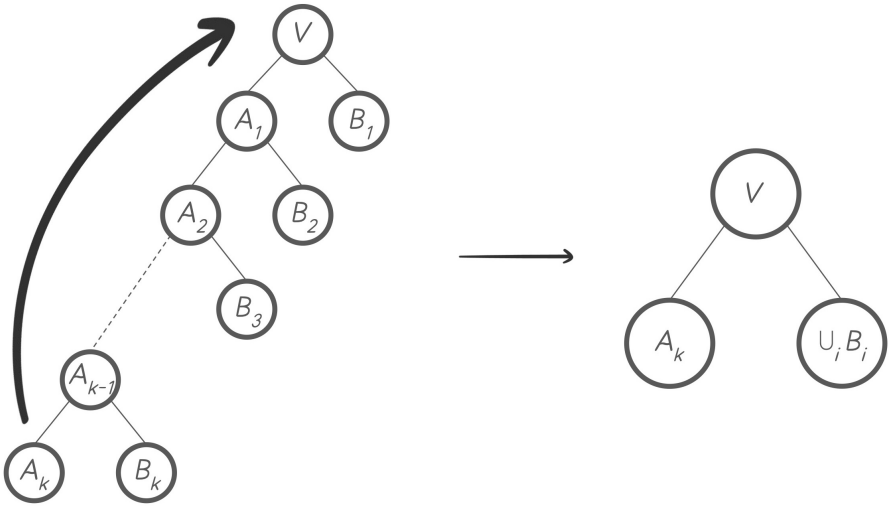


Figure 6.4: An illustration of one iteration in the recursion of RebalanceTree. We stop at depth k when $|B_k| \geq n/3$. The final partition is $A_k, B = \cup_{i=1}^{k} B_i$.

Lemma 6.4.3. *In Algorithm 8, every edge is separated by at most one tree rebalance operator.*

Finally, the operation cost of the rebalance operators comes from our termination threshold.

Lemma 6.4.4. *In Algorithm 8, every tree rebalance operator has operation cost at most* $3/2$.

In Theorem 6.4.1, the relative balance comes from Lemma 6.4.2, the operator properties come from Lemmas 6.4.3 and 6.4.4 respectively, and the approximation factor comes from Lemma 6.4.3 and Lemma 6.4.4 together.

6.4.2 Refining Relatively Rebalanced Trees

We now propose a significant extension of Algorithm 8 that allows us to get a stronger balance guarantee. Specifically (where ϵ may be a function of n):

Theorem 6.4.5. *Given a γ-approximation for cost, we can construct a $\frac{9\gamma}{2\epsilon}$-approximation for cost which guarantees ϵ-relative balance for $0 < \epsilon \leq 1/6$. The procedure modifies the resultant tree*

of Theorem 6.4.1 by applying subtree insertion and deletion operations of operation cost $\frac{3}{\epsilon}$, and separates every edge by at most one such operator.

To achieve this, we first apply the RebalanceTree operation. Then, at each split starting from the root, we invoke the SubtreeSearch procedure (Algorithm 11 in Section 6.6.2.2). This procedure identifies a small subtree located below the right child, deletes it, and repositions it under the left child to improve the relative balance of the hierarchy. By iterating this process sufficiently, we can reduce the relative balance to at most ϵ. The full procedure is formalized in Algorithm 9, which we refer to as RefineRebalanceTree.

Algorithm 9 RefineRebalanceTree

Require: A $\frac{1}{6}$-relatively balanced hierarchy tree T of size n, with bigger cluster always on the left, and balance parameter $\epsilon \in (0, 1/6)$.
Ensure: An ϵ-relatively balanced tree.

if $\epsilon \leq 1/(2n)$ **then**
    Return T
end if
$v = \text{root}(T)$
Let $T_{big} = T[\text{left}_T(v)]$, $T_{small} = T[\text{right}_T(v)]$
while $|\text{leaves}(T_{big})| \geq (1/2 + \epsilon)n$ **do**
    $\delta \leftarrow (|\text{leaves}(T_{big})| - n/2)/n$
    Let $T_{big} = \text{SubtreeSearch}(T_{big}, \delta n)$
end while
$T_{big} \leftarrow \text{RefineRebalanceTree}(T_{big}, \epsilon)$
$T_{small} \leftarrow \text{RefineRebalanceTree}(T_{small}, \epsilon)$
Return T' with root r with $\text{left}(r) = \text{root}(T_{big})$ and $\text{right}(r) = \text{root}(T_{small})$

The SubtreeSearch subroutine guarantees a proportional $2/3$ reduction in relative balance (see Section 6.6.2.2). With enough executions of SubtreeSearch, this makes the split ϵ-relatively balanced. Recursing down the tree then guarantees that the entire tree is ϵ-relatively balanced.

Lemma 6.4.6. *Algorithm 9 produces an ϵ-rebalanced tree.*

To bound the operators on top of those used by Algorithm 8, note that we only apply the

subtree deletion and insertion operators. Additionally, each edge cannot be separated more than once.

Lemma 6.4.7. *In Algorithm 9, every edge is separated by at most one subtree deletion and insertion operator.*

Finally, by limiting the depth of the $\mathrm{SubtreeSearch}$ function, the input parameter will always be at least ϵn. Thus, we can upper bound the operation cost of the deletion and insertion operators as follows.

Lemma 6.4.8. *In Algorithm 9, every subtree deletion and insertion operator has operation cost at most* $3/\epsilon$.

For Theorem 6.4.5, the relative balance comes from Lemma 6.4.6, the operator properties com from Lemmas 6.4.7 and 6.4.8 respectively, and the approximation factor comes from Lemma 6.6.2, Lemma 6.4.7, and Lemma 6.4.8 together.

6.4.3 Stochastically Fair Hierarchical Clustering

At this stage, we have nearly all the necessary tools to address the problem of stochastically fair hierarchical clustering. However, a final step involves applying level abstraction (Definition 6.3.6). To accomplish this, we introduce the $\mathrm{StochasticallyFairHC}$ operation, which enforces a single level abstraction by modifying the hierarchy to be $T' = T.\mathrm{abstract}(t, h_{\max})$ where t is a parameter, and $h_{\max}$ denotes the maximum depth of T. Note that, for this operation to work, the input hierarchy must be relatively balanced.

Theorem 6.4.9. *Given a* γ*-approximation for* cost*, let* $\epsilon = \frac{1}{c \log_2 n}$*, where* $c, \lambda \in O(1)$ *and* $\delta \in (0, 1)$*. Then in the stochastic fairness setting with* $\frac{1}{1-\delta}\alpha_\ell \leq p_\ell(v) \leq \frac{1}{1+\delta}\beta_\ell$ *for all* $v \in V$

and $\ell \in [\lambda]$, there is a $e^{4/(c(1-o(1))} \cdot \frac{3(1-\delta)\ln(cn)}{\delta^2 \min_{\ell\in[\lambda]} \alpha_\ell} \cdot \frac{9\gamma}{2\epsilon}$ fair approximation that succeeds with high probability. On top of the operators of Theorem 6.4.5, it only modifies the tree by applying one level abstraction of operation cost at most $e^{4/(c(1-o(1))} \cdot \frac{3\ln(cn)}{a\delta^2}$.

Note that for constant α_ℓ for all ℓ and δ, Theorem 6.4.9 yields an $O(\gamma \log n)$ approximation. Using the current state-of-the-art $\gamma = O(\sqrt{\log n})$ [Charikar and Chatziafratis, 2017, Dasgupta, 2016], this becomes $O(\log^{3/2} n)$—a substantial improvement on the prior best (albeit, deterministic) poly(n) fair approximation [Ahmadian et al., 2020a]. We note that, δ exhibits an important tradeoff: increasing δ increases the success probability, but diminishes the range of acceptable $p_\ell(v)$ values.

It might be tempting to suggest applying StochasticallyFairHC to any existing hierarchy, as opposed to one that is ϵ-relatively balanced. However, consider the instance with a highly unbalanced tree where all internal nodes have at least one leaf-child. The algorithm here would only merge the bottom t internal nodes into a single cluster, thereby not guaranteeing fairness. Adversarial constructions such as this demonstrate why the rebalancing process is important.

To bound the operation cost of the singular level abstraction, we explore the relative size of clusters at a specified depth in the hierarchy. The following guarantee is achieved by considering a root-to-vertex path where we always travel to maximally/minimally sized clusters according to the tree's relative balance.

Lemma 6.4.10. *Let T be an ϵ-relatively balanced tree, and let u and v be internal nodes at depth i in T. Then we must have that*

$$(1/2 - \epsilon)^i n \leq n_T(u), n_T(v) \leq (1/2 + \epsilon)^i n.$$

This equivalently implies $\frac{n_T(u)}{n_T(v)} \leq \frac{(1+2\epsilon)^i}{(1-2\epsilon)^i}$. *Furthermore, if* $i \leq \log_{1/2-\epsilon}(x/n)$ *for some arbitrary* $x \geq 1$ *and* $\epsilon = 1/(c \log_2 n)$ *defined by a constant* c*, then the maximum cluster size is at most* $e^{4/(c(1-o(1)))}x$.

Using this upper bound on the size of clusters at certain depths, we naturally obtain our operation cost.

Lemma 6.4.11. *In* StochasticallyFairHC*, the level abstraction has operation cost at most* $(1/2+\epsilon)^t n$.

To establish our fairness results, we first employ a Chernoff bound argument. Consequently, it is necessary to ensure that all internal nodes contain a sufficiently large number of elements. This requirement is satisfied through our established bounds on cluster sizes.

Lemma 6.4.12. *In* StochasticallyFairHC*, for any internal node* v, $n_{T'}(v) \geq (1/2-\epsilon)^t n$.

Finally, since union of two fair clusters is fair, we need only show that the property holds for the clusters at height 1 in the hierarchy. This, again, comes from a Chernoff bound.

Lemma 6.4.13. *Let* T' *be the tree obtained from applying* StochasticallyFairHC *with*

$$t = \log_{1/2-\epsilon}\left(\frac{3(1-\delta)\ln(cn)}{a\delta^2 n}\right),$$

where $a = \min_{\ell \in [\lambda]} \alpha_\ell$ *and* $\delta > 0$. *Then, with high probability,* T' *is stochastically fair for the given parameters* α_ℓ *and* β_ℓ *for all colors* $\ell \in [\lambda]$*, provided that*

$$\frac{1}{1-\delta}\alpha_\ell \leq p_\ell(v) \leq \frac{1}{1+\delta}\beta_\ell$$

for all $v \in V$ *and* $\ell \in [\lambda]$*, where* $\lambda = O(1)$.

This is sufficient to show Theorem 6.4.9. The fairness is a result of Lemma 6.4.13, the operator properties are a result of Lemma 6.4.11 and the obvious fact that we only apply one level abstraction, and the approximation factor comes from Lemma 6.6.3 and Lemma 6.4.11 together.

6.4.4 Deterministically Fair Hierarchical Clustering

Finally, we present our main results on the standard, *deterministic*, fair hierarchical clustering problem. This algorithm builds on top of the results from Theorem 6.4.5 and uses methods similar to Theorem 6.4.9. Stemming from previous algorithms, the procedure uses multiple level abstractions and introduces tree folding.

Theorem 6.4.14. *Given a* γ*-approximation for* cost *over* $\nu(V, \ell) = c_\ell n = O(n)$ *vertices for each color* $\ell \in [\lambda]$*, with* $h = n^\delta$ *for any constants* c, δ*, and* k*, there exists an algorithm that constructs a hierarchy* T' *such that:*

1. *Is a* $e^{\frac{4\log_2 n}{c(1-o(1))\lambda \log_2 h} + \frac{2}{c} + \frac{4}{c(1-o(1))}} \cdot \frac{9c^2\gamma}{4} \cdot n^\delta \log_2^2 n$*-approximation for cost.*

2. *Is fair for any parameters which guarantee that for all* $\ell \in [\lambda]$*:*

$$\beta_\ell \geq c_\ell \left(\frac{e^{4/c}}{kc_\ell} + e^{6/c} \right)^{1/\delta} \text{ and } \alpha_\ell \leq \frac{c_\ell}{e^{(6/c)\log_h(n)}}.$$

On top of the operations of Theorem 6.4.9, the procedure only modifies the tree by applying level abstractions and tree folding on k *subtrees with operation costs* $e^{2/c} n^\delta$ *and* $ke^{4/(c(1-o(1))}$

respectively. Each edge is separated in at most one level abstraction operator and in at most $\frac{\lambda}{\delta}$ *tree fold operators. Furthermore, this algorithm runs in* $O(n^2 \log n)$ *time.*

For $\gamma = O(\sqrt{\log n})$, this becomes $O(n^\delta \log^{5/2} n)$ for any constant c, $\delta \in (0, 1)$, and k which greatly improves the previous $O(n^{5/6} \log^{5/4}(n))$-approximation [Ahmadian et al., 2020a]. While we do not guarantee exact color ratio preservation as the previous work does, our algorithms: work for $O(1)$ colors, no longer require the ratio between colors points in the input to be exactly 1, and can get arbitrarily close to exact ratio preservation through parameterization.

In examining the fairness guarantee, note that all of the variables here are parameterizeable constants. Increasing k, c, and δ will all make these values get closer to the true proportions of the colors in the overall dataset, and this can be done to an arbitrary extent. Therefore, based off the parameterization, this allows us to enforce clusters to have nearly the same color proportions as the underlying dataset.

The algorithm intuitively works by recursively abstracting away the top $\log_2 h$ depth of the tree, where we end up setting $h = n^\delta$. Each time we do this, we produce "frontier clustering", or an h-sized clustering whose parents in the tree are all the root after level abstraction. Since the subtrees rooted at each cluster have the same topology (besides their leaves), we can then execute tree folding on any subset of them. We select cluster subtrees to fold together such that, once we merge the appropriate clusters, the clustering at this level will be more fair. Then, as we recurse down the tree, we subsequently either eliminate clusters (via level abstraction) or fold them to guarantee fairness. For a more detailed description, see Algorithm 10.

Algorithm 10 FairHC

Require: An $\epsilon = 1/(c \log_2 n)$ relatively balanced hierarchy tree T of size n on red and blue points, and parameters $h = 2^i$ and $k = 2^j$ for some $0 < j < i < \log_{1/2-\epsilon}(1/(2n\epsilon))$

Ensure: A fair tree.

Let $T \leftarrow T.\text{abstract}(0, i)$
if T is height 1 **then**
    Return T
end if
Let $\mathcal{V}$ be the children of $\text{root}(T)$
for each color $\ell \in [\lambda]$ **do**
    Order $\mathcal{V} = \{v_i\}_{i \in [h]}$ decreasing by $\frac{\ell(\text{leaves}(v_i))}{|\text{leaves}(v_i)|}$
    For all $i \in [k]$, $T \leftarrow T.\text{fold}(\{T'[v_{i+(j-1)k}] : j \in [h/k]\})$
end for
for each child v of $\text{root}(T)$ **do**
    Replace $T[v] \leftarrow \text{FairHC}(T[v])$
end for
Return T'

To see why this creates a fair, low-cost hierarchy, we first bound the metrics on the operators used. When executing level abstraction, we then leverage the relative balance and Lemma 6.4.10 to show that during FairHC we have a bound on the abstraction operation cost.

Lemma 6.4.15. *In Algorithm 10, the level abstraction has operation cost at most* $e^{2/c}h$.

Since any two vertices that are folded together must be at similar depths, the balance of our tree leads directly to a bound on the tree folding operation cost.

Lemma 6.4.16. *In Algorithm 10, each tree folding has operation cost at most* $ke^{4/(c(1-o(1))}$ *and acts on* k *trees.*

In order to bound the overall cost, we must now count the number of times an edge will be separated. Crucially for this argument, an edge that is separated by level abstraction can no longer be separated on a subsequent recursive step, and the number of tree fold operators is proportionally bounded by the recursive depth, as it only happens λ times each step.

Lemma 6.4.17. *In Algorithm 10, an edge e is separated by at most 1 level abstraction and $\lambda \log_2(n)/\log_2(h)$ tree folds. The maximum recursion depth is also at most $\log_2(n)/\log_2(h)$.*

The fairness guarantee is obtained through the ordering on Line 7 and the fold operation on Line 8 of Algorithm 10. One recursive step of FairHC incurs a small constant factor proportionate loss in potential fairness, and the number of times this loss occurs is bounded by the depth of recursion. To force these fractions to be arbitrarily close to the true color proportions, we simply tune the parameters c, k, and h.

Lemma 6.4.18. *Given an ϵ-relatively balanced hierarchy T over $\nu(V, \ell) = c_\ell n = O(n)$ vertices of any color $\ell \in [\lambda]$, Algorithm 10 yields a hierarchical clustering T' such that the amount of each color in each cluster (represented by vertex v) is bounded as follows:*

$$\frac{c_\ell}{e^{2\log_h n/c}} \leq \frac{\nu(\text{cluster}(v), \ell)}{n_T(v)} \leq c_\ell \cdot (e^{4/c}/(kc_\ell) + e^{6/c})^{\log_h n}.$$

In Theorem 6.4.14, fairness is a result of Lemma 6.4.18, the operator properties are a result of Lemmas 6.4.15, 6.4.16, and 6.4.17, and the approximation factor has already been worked out by Lemma 6.6.9.

6.5 Experiments

This section validates our algorithms from Section 6.4. These simulations demonstrate that our algorithm incurs only a modest loss in the hierarchical clustering objective, while improving the fairness of every cluster. Specifically, we present the approximate cost increases as a function of Algorithm 10's defining parameters: c, δ, and k.

Datasets. We use two data sets, *Census* and *Bank*, from the UCI data repository [Dua and Graff, 2017]. Within each, we subsample only the features with numerical values. To compute the *cost* of a hierarchical clustering we set the similarity to be $w(i,j) = \frac{1}{1+d(i,j)}$ where $d(i,j)$ is the Euclidean distance between points i and j. We color data based on binary (represented as blue and red) protected features: *race* for *Census* and *marital status* for *Bank* (both in line with the prior work of [Ahmadian et al., 2020a]). As a result, *Census* has a blue to red ratio of 1:7 while *Bank* has 1:3.

We then subsample each color in each data set such that we retain (approximately) the data's original balance. We use samples of size 256. For each experiment, we do 10 replications and report the average results. We vary the parameters $c \in \{2^i\}_{i=0}^{5}, \delta \in (\frac{1}{8}, \frac{7}{8})$, and $k \in \{2^i\}_{i=1}^{4}$ to experimentally validate their theoretical impact on the approximate guarantees of Section 6.4.

Implementation. The Python code for the following experiments are available in the Supplementary Material. We start by running average-linkage, a popular hierarchical clustering algorithm. We then apply Algorithms 8 - 10 to modify this structure and induce a *fair* hierarchical clustering that exhibits a mild increase in the cost objective.

Metrics. In our results we track the approximate cost objective increase as follows: Let G be our given graph, T be average-linkage's output, and T' be Algorithm 10's output. We then measure the ratio $\text{RATIO}_{cost} = \frac{cost_G(T')}{cost_G(T)}$.

Results.. We first note that the average-linkage algorithm must construct unfair trees since, for each data set, the algorithm induces some monochromatic clusters. Thus, our resultant fair clustering is of considerable value in practice.

In Figure 1, we plot the change in cost ratio as the parameters (c, δ, k) are varied for the two

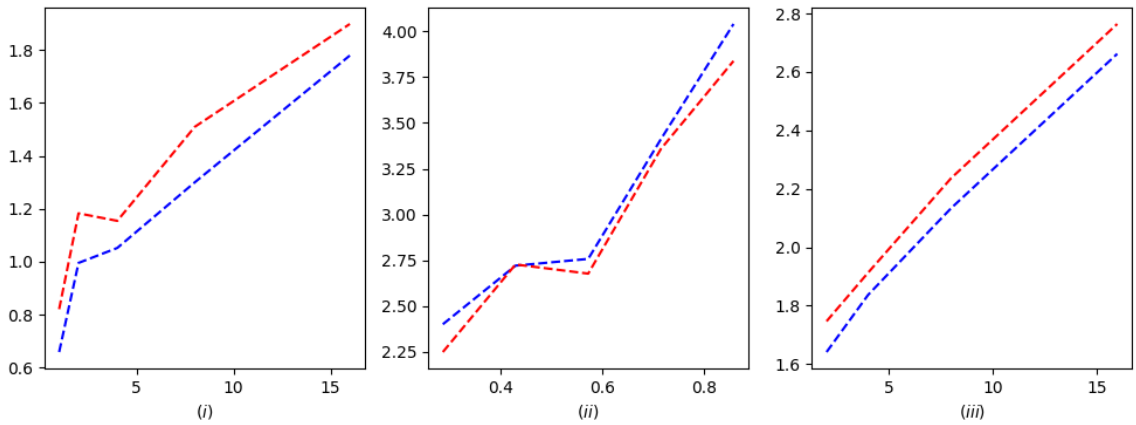


Figure 6.5: Cost ratio of Algorithm 10 as compared to average-linkage. (i) Ratio increase as a function of the parameter c, (ii) ratio increase as a function of the parameter δ, and (iii) ratio increase as a function of k. Blue lines indicate the result for *Census* dataset whereas red indicates the *Bank* dataset results.

datasets. Supporting our theoretical results, increasing our fairness parameters leads to a modest increase in cost. This is an empirical illustration of our fairness-cost approximation tradeoff according to our parameterization. Note that the results are consistent across tested datasets.

We additionally illustrate the resulting balance of our hierarchical clustering algorithm by presenting the distribution of the cluster ratios of the projected features (blue to red points) in Figure 6.6 for the *Census* data. The output of average-linkage naturally yields an unfair clustering of the data, yet after applying our algorithm on top this hierarchy we see that the cluster's balance move to concentrate about the underlying data balance of 1:7. An equivalent figure for the *Bank* dataset is provided in Figure 6.7.

6.6 Proofs

We here formally verify all theorems and their intermediary lemmas.

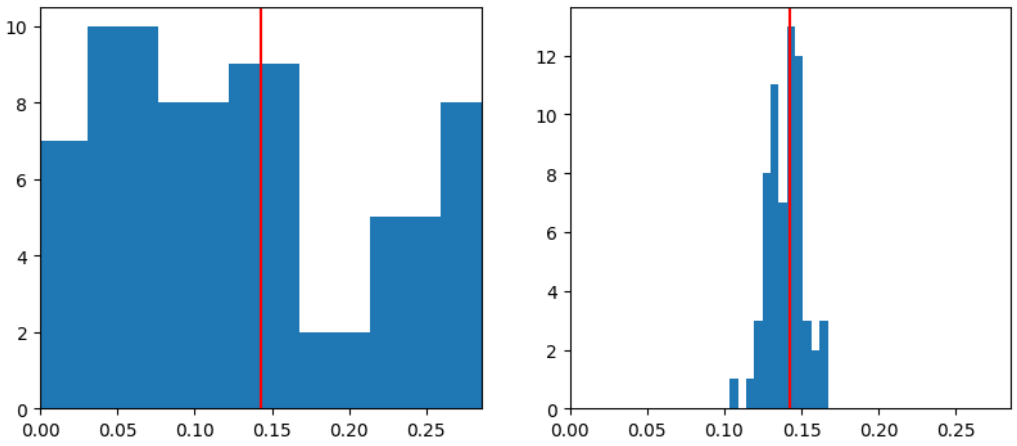


Figure 6.6: Histogram of cluster balances after tree manipulation by Algorithm 10. The left plot depicts the balances after applying the average- linkage algorithm and the right shows the result of applying our algorithm. The vertical red line indicates the balance of the dataset. Parameters were set to $c = 4, \delta = \frac{3}{8}, k = 4$ for the above clustering result.

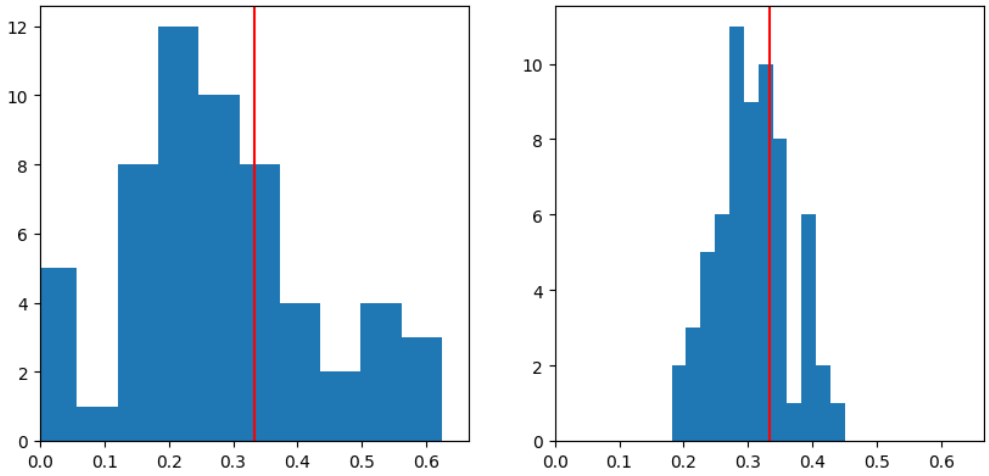


Figure 6.7: Histogram of cluster balances after tree manipulation by Algorithm 10. The left plot depicts the balances after applying the average- linkage algorithm and the right shows the result of applying our algorithm. The vertical red line indicates the balance of the dataset. Parameters were set to $c = 4, \delta = \frac{3}{8}, k = 4$ for the above clustering result.

6.6.1 Tree Properties and Operators

Here we present all our proofs and theoretical results regarding our tree operator properties.

We start by discussing our tree rebalance operator. Effectively, any edge whose end points are

separated by a tree rebalance operator was contained in a cluster of size $n_T(u)$, and now we guarantee they are in a cluster of size $n_T(v)$.

Lemma 6.6.1. *Let T be a tree, and define $T' = \text{tree\_rebalance}(u, v)$ for a node $u \in V$ and its ancestor v. The only edges affected by this operation are those of the form $e = (x, y)$ with $x \in \text{cluster}(u)$ and $y \in \text{cluster}(a) \setminus \text{cluster}(u)$, where a is the child of v containing the edge. Moreover, the operation cost is bounded above by*

$$\Delta = \frac{n_T(v)}{n_T(p)},$$

where p is the parent of u.

Proof. Let $e = (x, y)$ be an edge separated by the $\text{tree\_rebalance}(u, v)$ for internal nodes u and v. Traversing the path from v to u, we label the internal nodes we visit as $\{A_i\}_{i=1}^{k-1}$ and their sibling, unvisited, nodes as $\{B_i\}_{i=1}^{k-1}$. Upon reaching the desired node u, we let $A_k = \text{cluster}(u)$ be the set of vertices corresponding to the internal node u and B_k its sibling.

By definition of the rebalance operation, we now split the set of vertices of $\text{cluster}(v)$ into A_k and $\cup_{i=1}^{k} B_i$. Thus, for e to be separated by this split, we must have that one endpoint is in A_k and the other is in $\cup_{i=1}^{k} B_i$. Without loss of generality, assume $x \in \text{cluster}(u)$ and $y \in \text{cluster}(a) \setminus \text{cluster}(u)$ where a is the child of v containing e. This further implies that the lowest common ancestor (LCA) of x and y was on the path from v to u (excluding u). The smallest such cluster is the immediate parent of u, denoted as p. Therefore, we must have that $n_T(e) \geq n_T(p)$.

Lastly, note that, after executing the operation, the LCA of x and y in T' is v (whose total number of children has not changed). Therefore, $n_{T'}(e) = n_T(v)$. Putting this all together, we

have that $\text{cost}_{T'}(e) \leq \frac{n_T(v)}{n_T(p)} \cdot \text{cost}_T(e)$, completing the result. □

For our subtree deletion and insertion, the idea is that an edge that is separated costs at least $n_T(v)$ in the original tree, but may cost up to $n_T(u \wedge v)$ in the modified tree.

Lemma 6.6.2. *Given a tree T, let $T' = \text{del\_ins}(u, v)$ for two nodes u and v, where u is not an ancestor of v. The only edges separated by this are $e = (x, y)$ such that $x \in \text{cluster}(u)$ and $y \in \text{cluster}(u \wedge v) \setminus \text{cluster}(u)$. The operation cost is bounded above by $\Delta = n_T(u \wedge v)/n_T(u)$.*

Proof. Let $e = (x, y)$ be an edge separated by the $\text{del\_ins}(u, v)$ for internal nodes u and v. We first note that at least one end point must be in $\text{cluster}(u)$ since the only nodes moved are contained in the corresponding subtree rooted at u. Without loss of generality, assume $x \in \text{cluster}(u)$. Since e is separated, their LCA must changed. Therefore, $y \notin \text{cluster}(u)$. Lastly, observe that if the LCA is higher than $u \wedge v$ in the tree, then the smallest containing cluster for the edge will not change. We must then have that $y \in \text{cluster}(u \wedge v) \setminus \text{cluster}(u)$.

From the above argument, we know that in the original tree the LCA of x and y must be higher in the tree than u. This gives us $n_T(e) \geq n_T(u)$. Lastly, we must have that the internal node corresponding to $u \wedge v$ in the original tree must contain exactly the same points after applying the operation. Thus, after applying the operation, the new LCA of x and y must be $u \wedge v$ in the worst case. Therefore, $n_{T'}(e) \leq n_T(u \wedge v)$. Combining these inequalities gives the result. □

The level abstraction operator is inherently more complex than other operations, as it modifies entire levels of the hierarchy rather than individual splits. However, we can still apply the notion of operation cost to quantify its impact.

To establish an upper bound on the cost induced by this operator, we analyze the largest and smallest clusters across depths h_1 to h_2 in T. Specifically, the worst-case cost inflation is

determined by the ratio of the largest cluster at depth h_1 to the smallest cluster at depth h_2.

Lemma 6.6.3. *Let T' be the hierarchy obtained by applying the level abstraction operator to hierarchy T between heights h_1 and h_2 (where $h_1 < h_2$). An edge $e = (x, y)$ is separated by this operation if and only if the least common ancestor of x and y in T lies between depths h_1 and h_2. The operation cost of this level abstraction is bounded above as*

$$\Delta \leq \max_{(u,v)\in S} \frac{n_T(u)}{n_T(v)},$$

where S is the set of all pairs of clusters (u, v) that are abstracted away in the transformation, and the maximum is taken over all such pairs.

Proof. First, observe that if the LCA of x and y in T is below h_1 or above h_2, then the points contained by this internal node are not altered by the operation. Thus, we restrict attention to those whose ancestor is within this range of depths, and therefore separated by the operation.

For a separated edge $e = (x, y)$, we must have that the LCA (which is at depth between h_1 and h_2) is contracted to its ancestor at depth h_1. Let V_h be the vertices at depth h_1 to h_2 in the tree, and further denote $v^* = \operatorname{argmin}_{v\in V_h} n_T(v)$. By definition, we must have that $n_T(e) \geq n_T(v^*)$. Furthermore, the ancestor the LCA is contracted into must be at depth h_1. The size of this cluster does not change since no points are added or removed, only the tree structure below it is altered. Let $u^* = \operatorname{argmax}_{u\in V_h} n_T(u)$. Then, we must have that after the abstraction, $n_{T'}(x \wedge y) \leq n_T(u^*)$. Thus, u^* and v^* are exactly the pair that maximize the ratio in the Lemma statement, and we must further have that $\operatorname{cost}_{T'}(e) \leq \frac{n_T(u^*)}{n_T(v^*)} \cdot \operatorname{cost}_T(e)$. □

The tree folding operation introduces additional complexity, as it involves merging multiple

clusters into a single structure. Unlike simpler operations that modify individual splits, tree folding must account for both the variation in cluster sizes and the parameter k, which represents the number of clusters being merged.

To bound the proportional increase in cost induced by this operation, we consider the ratio of cluster sizes before and after folding. Specifically, the worst-case cost inflation is determined by the product of two factors: the maximum ratio between the largest and smallest clusters involved in the merge and the factor k, which accounts for the cumulative effect of merging multiple clusters.

Lemma 6.6.4. *Let T' be the hierarchy obtained by applying the tree folding operator on hierarchy T. The operation cost of this folding is bounded above as*

$$\Delta \leq k \cdot \max_{(u,v)\in S} \frac{n_T(u)}{n_T(v)}$$

where S is the set of all pairs of clusters (u, v) which are mapped onto each other by the operation.

Proof. Let $e = (x, y)$ be an edge separated by the tree fold operator on trees $T_1, ..., T_k$. Assume that the LCA, $x \wedge y$, is not contained in any of the T_i. Then, the size of the cluster rooted at $x \wedge y$ does not change by the operation, so we can proceed on the assumption that any separated edge must have $x \wedge y \in T_i$ for some $i \in [k]$. Without loss of generality, assume $x \wedge y \in T_1$.

Let ϕ_1 be the bijection of internal nodes of T_1 into the folded tree, T_f. Crucially, observe that any leaf node of T_i (for $i \neq 1$) will be an ancestor of $\phi_1(x \wedge y)$ if and only if it has an ancestor,

a, such that $\phi_i(a) = \phi_1(x \wedge y)$. We then have that

$$n_{T'}(x \wedge y) = n_{T'}(\phi_1(x \wedge y)) \leq \sum_{i \in [k]} n_{T'}(\phi_i^{-1}(\phi_1(x \wedge y))$$

Now, let $u^* = \max\{n_T(u) : u \in T_i, i \in [k], \phi_i(u) = \phi_1(x \wedge y)\}$. We can now rewrite

$$\begin{aligned} n_{T'}(x \wedge y) &\leq \sum_{i \in [k]} n_{T'}(\phi_i^{-1}(\phi_1(x \wedge y)) \\ &\leq \sum_{i \in [k]} n_T(u) \\ &= k n_T(u). \end{aligned}$$

Therefore, we have that $\text{cost}_{T'}(e) \leq w(e) \cdot$. Combining this with the fact that $\text{cost}_T(e) \geq w(e) \cdot n_T(v)$, where v is the smallest node containing e before folding, gives the result. □

6.6.2 Algorithmic Approximation Guarantees

In this section, we prove all lemmas, theorems, and missing algorithmic discussion regarding our main results.

6.6.2.1 RebalanceTree

This section contains the proofs regarding RebalanceTree.

Proof of Lemma 6.4.2. By our definition of A_i and B_i for all $i \in [k]$, we have that $|A_{k-1}| \geq \frac{2n}{3}$. Moreover, since $|A_i| > |B_i|$ at each i, we must have that $|A_k| \geq \frac{1}{2}|A_{k-1}| \geq \frac{n}{3}$. By the while loop condition, we further have that $A_k < \frac{2n}{3}$ and $B_k \geq \frac{n}{3}$. Therefore, $\frac{n}{3} \leq |A_k|, |\cup_{i \in [k]} B_i| \leq \frac{2n}{3}$.

By constructing the initial split in this manner, the first application of the tree rebalancing operation ensures that the resulting partition satisfies the relatively balanced condition. This property is then preserved recursively at each subsequent level, as each successive split maintains the balance criterion. Consequently, by induction, the entire tree remains $\frac{1}{6}$-relatively balanced.

□

Proof of Lemma 6.4.3. Consider an edge $e = (x, y)$ that is first rebalanced at some recursive step in Algorithm 8. By Lemma 6.6.1, once this operation is applied, x and y must be separated at the root of the current tree. Consequently, in any subsequent recursive step, at most one endpoint of e will remain in the subtree being processed, ensuring that e cannot be separated again. □

Proof of Lemma 6.4.4. The rebalance operator is applied to v (the node found) at r. Notice that v's parent p must be such that $n_T(p) \geq \frac{2}{3} n_T(r)$, otherwise the loop would have stopped earlier. Therefore, by Lemma 6.6.1, the operation cost is $\frac{n_T(r)}{n_T(p)} \leq \frac{3}{2}$. □

Proof of Theorem 6.4.1. Let T^* be the optimal tree, let T_1 be our guaranteed γ-approximation on T, and let T' be our output. By Lemma 6.4.2, T' is $1/6$-relatively balanced. By Lemmas 6.4.3 and 6.4.4, every edge is separated by at most one tree rebalance operator of length at most $3/2$. Because of this, $\text{cost}_{T'}(e) \leq (3/2)\text{cost}_{T_1}(e)$. Summing over all edges yields $\text{cost}(T) \leq (3/2)\text{cost}(T_1) \leq \gamma \text{cost}(T^*)$. □

6.6.2.2 RefineRebalanceTree

This section contains the proofs regarding RefineRebalanceTree, as well as the algorithmic description of SubtreeSearch.

As discussed in the body, at a given split, SubtreeSearch traverses the tree below the larger cluster further down in a similar manner until we find a sufficiently small cluster. This cluster must be smaller than the current balance error, ϵ. We simply do this by always traversing to the larger cluster as in Algorithm 11 until its smaller child is sufficiently small. We then remove that subtree, traverse back to the top of the tree, and try to reinsert the subtree by recursing down the right children.

Algorithm 11 SubtreeSearch

Require: A $\frac{1}{6}$-relatively balanced hierarchy tree T of size n, with smaller cluster always on the left and error parameter s.

Ensure: Modified $\frac{1}{6}$-relatively balanced T by a subtree deletion and insertion of a subtree of size between $s/3$ and s

$v = \text{root}(T)$
while $|\text{leaves}(\text{left}_T(v))| > s$ **do**
    $v \leftarrow \text{right}_T(v)$
end while
$v \leftarrow \text{left}_T(v)$
$u \leftarrow \text{root}(T)$
while $|\text{leaves}(\text{right}_T(u))| \geq |\text{leaves}(v)|$ **do**
    $u \leftarrow \text{left}_T(u)$
end while
$T' \leftarrow T.\text{del\_ins}(u, v)$
Return T

Lemma 6.6.5. SubtreeSearch *preserves* $\frac{1}{6}$*-relative balance.*

Proof of Lemma 6.6.5. Consider T, the initial tree at the start of the algorithm, and let v be the vertex whose subtree is relocated. First, we analyze the effect of deleting v's subtree before considering its reinsertion. The only vertices whose corresponding cluster sizes are affected—specifically, reduced—are the ancestors of v. Notably, each of these ancestors is a right child (i.e., the larger sibling at the start of the process), and their sizes decrease by exactly $n_T(v)$.

Let p be the parent of v. Since $v = \text{left}_T(p)$, by definition of the left child, we have

$n_T(v) \leq \frac{1}{2} n_T(p)$. Since deleting v's subtree removes exactly $n_T(v)$ vertices from p, the size of p is at worst halved, $n_{T'}(p) \geq \frac{1}{2} n_T(p)$. Let q be the sibling of p. Since p is a right child, we initially have $n_T(p) \geq n_T(q)$. After deletion, this ensures that

$$\begin{aligned} n_{T'}(p) &\geq \frac{1}{2} n_T(p) \\ &\geq \frac{1}{2} n_T(q) \\ &= \frac{1}{2} n_{T'}(q) \end{aligned}$$

where the last equality holds since q is not an ancestor of v. Thus, the final clusters at this split remain within a ratio of $1/3$ to $2/3$ relative to their parent, preserving the relative balance condition.

For any ancestor node a of p in T, the same argument holds: since $n_T(a) > n_T(p)$ both before and after deletion, and a is also a right child, its balance remains unaffected. Therefore, the entire tree remains $\frac{1}{6}$-relatively balanced after the subtree deletion.

Now, we analyze the second half of the algorithm, where the subtree $T[v]$ is reinserted. Let u be the selected vertex where v is inserted, p be its new parent, g be its original parent (now its grandparent), and let $r = \mathrm{right}_T(g)$ be its original sibling.

Before insertion, the while-loop condition ensures that $n_T(r) \geq n_T(v)$. Since the tree remains $\frac{1}{6}$-relatively balanced and r is the sibling of u, we have $n_T(u) \geq \frac{1}{2} n_T(r)$. Furthermore, the fact that the algorithm did not stop at p guarantees that $n_T(r) \geq n_T(v)$, which implies $n_T(u) \geq \frac{1}{2} n_T(v)$. Since the algorithm terminates at u, we also know that $n_T(\mathrm{right}_T(u)) \leq n_T(v)$.

As $\text{right}_T(u)$ is the larger child of u, it follows that

$$n_T(u) \leq 2 \cdot n_T(\text{right}_T(u)) \leq 2 \cdot n_T(v).$$

Since v is now the sibling of u and their sizes satisfy

$$\frac{1}{2} n_T(v) \leq n_T(u) \leq 2 \cdot n_T(v),$$

the relative balance condition is maintained at this split.

The only other vertices impacted by the insertion are u's ancestors. For an ancestor node a of u in T, the argument also holds since $n_T(a) \geq n_T(p) \geq n_T(v)$ meaning $n_{T'}(a) \leq 2n_T(a)$ and a must be a left (and therefore smaller) child. Therefore, the relative balance is kept at all splits involving ancestors of u, thus we have relative balance. □

We further guarantee that the search procedure finds a subtree of size at least $s/2$ to move. This is implied from the first while loop's end condition.

Lemma 6.6.6. *In Algorithm 11,*

$$\frac{s}{3} \leq n_T(v) \leq s.$$

Proof of Lemma 6.6.6. When the while loop on Line 2 terminates, we have found the first visited vertex whose left child (which ends up being the final v) is of size at most s. Thus, $n_T(v) \leq s$. Since this was the first such instance, if g is the grandparent of v, this means $n_T(\text{left}_T(g)) > s$ since the loop continued after g. Due to the convention that right children are larger, and v's

parent p is $\text{right}_T(g)$, $n_T(p) \geq n_T(\text{left}_T(g)) > s$. Thus, we have $\frac{1}{6}$-relative balance,

$$n_T(v) \geq \frac{1}{3} n_T(p) \geq \frac{1}{3} s.$$

□

Proof of Lemma 6.4.6. At each iteration of Algorithm 9, as long as the relative balance exceeds ϵ, we relocate a subtree of size between $\frac{1}{3}\delta n$ and δn, as established by Lemma 6.6.6, where δ represents the current relative balance. This guarantees that the relative balance at the first split decreases by a factor of $\frac{2}{3}$, while Lemma 6.6.5 ensures that the remainder of the tree remains $\frac{1}{6}$-relatively balanced. The process continues until the relative balance of the first split falls below ϵ. Upon recursion, the tree retains its $\frac{1}{6}$-relative balance, ensuring that all sufficiently large splits satisfy the ϵ-relative balance condition. □

Proof of Lemma 6.4.7. Consider an edge e that is first separated by some subtree deletion and insertion operator at some recursive step in Algorithm 9. Notice e must now be separated at the current tree's root. This means that at any further level of recursion, only one of e's end points will be present, so it cannot be separated again. □

Proof of Lemma 6.4.8. The subtree deletion and insertion operator is applied at u of $T[v]$ when $u \wedge v$ is the root, i.e., $n_T(u \wedge v) = n_T(r) \leq n$ where r is the current tree's root and n is our original data set size. We never allow the algorithm to continue with $\delta \leq \epsilon$, therefore the smallest tree size $n_T(v)$ that we move is $\frac{n\epsilon}{3}$ by Lemma 6.6.6. Thus, the operation cost is at most

$$\frac{n_T(u \wedge v)}{n_T(v)} \leq \frac{3}{\epsilon}$$

by Lemma 6.6.2. □

Proof of Theorem 6.4.5. Let T^* be the optimal tree, let T_1 be the $\frac{1}{6}$-relatively balanced $\frac{3\gamma}{2}$-approximate solution guaranteed by Theorem 6.4.1, and let T' be the final output of our algorithm.

By Lemma 6.4.6, the tree T' is ϵ-relatively balanced. Furthermore, by Lemmas 6.4.7 and 6.4.8, each edge is separated by at most one subtree deletion and insertion operation, with an associated operation cost bounded above by $\frac{3}{\epsilon}$. Thus, by Lemma 6.6.2, we obtain the following bound for any edge e:

$$\text{cost}_{T'}(e) \leq \frac{3}{\epsilon}\text{cost}_{T_1}(e).$$

Summing over all edges, it follows that

$$\text{cost}(T') \leq \frac{3}{\epsilon}\text{cost}(T_1).$$

Finally, applying the approximation guarantee for T_1, we conclude that

$$\text{cost}(T') \leq \frac{3}{\epsilon} \cdot \frac{3\gamma}{2}\text{cost}(T^*) = \frac{9\gamma}{2\epsilon}\text{cost}(T^*).$$

□

6.6.2.3 StochasticallyFairHC

This section contains the proofs regarding StochasticallyFairHC.

Proof of Lemma 6.4.10. Since T is ϵ-relatively balanced, any cluster A that splits into clusters B and C satisfies $(1/2 - \epsilon)|A| \leq |C| \leq |B| \leq (1/2 + \epsilon)|A|$, without loss of generality. Now, note

at depth 0 we have only the root node which defines a cluster of size n. At depth 1, we must the have that the split into two clusters, A_1 and B_1, must have

$$(1/2 - \epsilon)n \leq |A_1| \leq |B_1| \leq (1/2 + \epsilon)n$$

Taking a path from the root to the maximally sized child at each step, then we guarantee that if p is a parent node of some w on the path: $n_T(u) \leq (\frac{1}{2} + \epsilon)n_T(p)$. Let the maximal cluster at depth $i - 1$ be further split into A_i and B_i Since we traverse i levels, we get for any i-level vertex u, $n_T(u) \leq (\frac{1}{2} + \epsilon)^i n$. By the reverse logic (i.e., traversing from the root to a minimally sized child), for any i-level vertex v, $n_T(v) \geq (\frac{1}{2} - \epsilon)^i n$. We can now directly observe that

$$\frac{n_T(u)}{n_T(v)} \leq \frac{(1 + 2\epsilon)^i}{(1 - 2\epsilon)^i}.$$

Now, assume $i \leq \log_{1/2-\epsilon}\left(\frac{x}{n}\right)$. By the above argument, we have that the smallest cluster size at level i is at least $(\frac{1}{2} - \epsilon)^i n$. Rearrange the ratio bound, we see

$$\begin{aligned} n_T(u) &\leq \left(\frac{1 + 2\epsilon}{1 - 2\epsilon}\right)^i n_T(v) \\ &\leq \left(\frac{1 + 2\epsilon}{1 - 2\epsilon}\right)^i \left(\frac{1}{2} - \epsilon\right)^i n \end{aligned}$$

Using the assumed upper-bound on i, this further yields

$$\begin{aligned} n_T(u) &\leq \left(\frac{1 + 2\epsilon}{1 - 2\epsilon}\right)^{\log_{1/2-\epsilon}(n/x)} (1/2 - \epsilon)^{\log_{1/2-\epsilon}(x/n)} n \\ &= \left(\frac{1 + 2\epsilon}{1 - 2\epsilon}\right)^{\log_{1/2-\epsilon}(n/x)} x. \end{aligned}$$

We further observe that

$$\begin{aligned}\frac{1+2\epsilon}{1-2\epsilon} &= 1 + \frac{4\epsilon}{1-2\epsilon} \\ &= 1 + \frac{4}{c(1-2\epsilon)\lg n}.\end{aligned}$$

Thus, the remaining term is at least 1. Therefore, to bound this exponential, we must upper bound the exponent as follows:

$$\begin{aligned}\log_{\frac{1}{2}-\epsilon}\left(\frac{x}{n}\right) &= \frac{\lg\left(\frac{x}{n}\right)}{\lg\left(\frac{1}{2}-\epsilon\right)} \\ &= \frac{\lg\left(\frac{n}{x}\right)}{\lg\left(\frac{1}{\frac{1}{2}-\epsilon}\right)} \\ &\leq \lg\left(\frac{n}{x}\right) && (\epsilon \in (0, \tfrac{1}{2})) \\ &\leq \lg(n) && (x \geq 1)\end{aligned}$$

Combining the above facts, we have that

$$n_T(u) \leq \left(1 + \frac{4}{c(1-2\epsilon)\lg n}\right)^{\lg n} x \leq x \cdot e^{4/c(1-o(1))}.$$

□

Proof of Lemma 6.4.11. By Lemma 6.4.10, the smallest cluster at depth $i \geq t$ is at most $(1/2 + \epsilon)^t n$. If we assume a trivial cluster size lower bound of 1, this implies for any contracted internal nodes u and v in the level abstraction, $n_T(u)/n_T(v) \leq (1/2+\epsilon)^t n$. □

Proof of Lemma 6.4.12. By Lemma 6.4.10, the largest cluster at depth $i \leq t$ in T is at least

$(1/2-\epsilon)^t n$. When we execute level abstraction, this cluster size is not changed, but we know all other potentially smaller clusters are contracted into their parents. Thus, this is the smallest cluster size in T'. □

Proof of Lemma 6.4.13. Consider a vertex v at height 1. By Lemma 6.4.12,

$$n_{T'}(v) \geq (1/2-\epsilon)^t n = \frac{3(1-\delta)}{a\delta^2}\ln(cn)$$

where the equality comes from assumption on t. Fix some $\ell \in [\lambda]$. Let $X_{\ell v}$ count the number of vertices of color ℓ in $\text{leaves}(v)$. Note that this quantity is a sum of Bernoulli trials, so $\mathbb{E}[X_{\ell v}] = \sum_{u\in\text{cluster}(v)} p_\ell(u)$. By the given constraint that $p_\ell(u) \geq \frac{1}{1-\delta}\alpha_\ell$ for all u, combined with Lemma 6.4.12, we obtain the following bounds:

$$\begin{aligned}
\mathbb{E}[X_{\ell v}] &= \sum_{u\in\text{cluster}(v)} p_\ell(u) \\
&\geq \sum_{u\in\text{cluster}(v)} \frac{1}{1-\delta}\alpha_\ell \\
&\geq \sum_{u\in\text{cluster}(v)} \frac{1}{1-\delta}a \\
&= n_{T'}(v)\frac{1}{1-\delta}a \\
&\geq \frac{3(1-\delta)}{a\delta^2}\ln(cn)\cdot\frac{1}{1-\delta}a \\
&= \frac{3}{\delta^2}\ln(cn)
\end{aligned}$$

Now, applying a Chernoff bound with error parameter δ, we obtain:

$$\begin{aligned}\Pr\left[\,|X_{\ell v} - \mathbb{E}\left[\,X_{\ell v}\,\right]| \geq \delta\mathbb{E}\left[\,X_{\ell v}\,\right]\right] &\leq 2\exp(-\mathbb{E}\left[\,X_{\ell v}\,\right]\delta^2/3) \\ &\leq 2\exp(-\frac{3}{\delta^2}\ln(cn)\delta^2/3) \\ &= \frac{2}{cn}\end{aligned}$$

Thus, with probability at least $1 - \frac{2}{cn}$:

$$\begin{aligned}X_{\ell v} - \mathbb{E}\left[\,X_{\ell v}\,\right] &\leq \delta\mathbb{E}\left[\,X_{\ell v}\,\right] \\ \iff X_{\ell v} &\leq (1+\delta)\mathbb{E}\left[\,X_{\ell v}\,\right] \\ &\leq (1+\delta)\cdot\frac{1}{1+\delta}\beta_\ell n_{T'}(v) \\ &\leq \beta_\ell n_{T'}(v)\end{aligned}$$

where the third inequality comes from the upper-bound parameters for $p_\ell(u)$. Thus, the cluster $\mathrm{leaves}(v)$ satisfies the upper bound stochastically fair constraint for color ℓ with high probability. By a symmetric derivation, we can show that the lower bound holds with the same high probability. Namely:

$$\begin{aligned}-X_{\ell v} + \mathbb{E}\left[\,X_{\ell v}\,\right] &\geq \delta\mathbb{E}\left[\,X_{\ell v}\,\right] \\ \iff X_{\ell v} &\geq (1-\delta)\mathbb{E}\left[\,X_{\ell v}\,\right] \\ &\geq (1-\delta)\cdot\frac{1}{1-\delta}\alpha_\ell n_{T'}(v) \\ &\geq \alpha_\ell n_{T'}(v).\end{aligned}$$

We now show that the result must further hold for all colors. Let n_1 be the number of internal nodes with only leaf-children (height 1). Observe that, since a and δ are assumed to be $O(1)$, we have that the minimum cluster size at height 1 (ie. containing only leaf-children) is $O(\log n)$. Therefore, we have that $n_1 = O\left(\frac{n}{\log n}\right)$. Further note that, if each of these vertices is fair, then we have that the overall clustering is fair since the union of fair clusters is, again, fair. Combining these facts, it suffices to show that the fairness constraint holds for all ℓ and height 1 nodes v. Taking a union bound over all λ colors and n_1 values for the v, we find that with probability $1 - O\left(\frac{\lambda n/\log n}{cn}\right) = 1 - O\left(\frac{1}{\log n}\right)$ all height 1 clusters must be fair. Thus, the hierarchy is fair by the union closure of fairness. □

We now combine the intermediary lemmas of this section to prove the main theorem.

Proof of Theorem 6.4.9. Let T^* be the optimal tree, let T_1 be our ϵ-relatively balanced $\frac{9\gamma}{2\epsilon}$ approximation guaranteed by Theorem 6.4.5, and let T' be our output. By Lemma 6.4.13, T' satisfies our fairness constraints. By Lemma 6.4.11 and the fact that we only apply one operator, every edge is separated by at most one level abstraction operator of operation cost at most $(1/2+\epsilon)^t n$. But, we further know from Lemma 6.4.10 that this is bounded above by $e^{4/(c(1-o(1))} \cdot \frac{3(1-\delta)\ln(cn)}{a\delta^2}$. As a result, $\text{cost}_{T'}(e) \leq e^{4/(c(1-o(1))} \cdot \frac{3(1-\delta)\ln(cn)}{a\delta^2}\text{cost}_{T_1}(e)$. Summing over all edges yields

$$\begin{aligned}\text{cost}(T) &\leq e^{4/(c(1-o(1))} \cdot \frac{3(1-\delta)\ln(cn)}{a\delta^2}\text{cost}(T_1) \\ &\leq e^{4/(c(1-o(1))} \cdot \frac{3(1-\delta)\ln(cn)}{a\delta^2} \cdot \frac{9\gamma}{2\epsilon}\text{cost}(T^*)\end{aligned}$$

□

6.6.2.4 FairHC

This section contains the proofs and additional theoretical discussion regarding FairHC.

Lemma 6.6.7. StochasticallyFairHC *with* $t = \log_{1/2-\epsilon}(1/(2n\epsilon))$ *outputs a hierarchy where the* ϵ*-relatively balanced guarantee holds for all splits except those forming the leaves. Additionally, it admits a proportional cost increase of at most* $\frac{1}{2}ce^{4/(c(1-o(1))} \log_2 n$.

Proof of Lemma 6.6.7. Let T be our, ϵ-relative balanced, input. Notice that StochasticallyFairHC only modifies T's structure below depth $\log_{1/2-\epsilon}(1/(2n\epsilon))$, which means any balance guarantees hold up to that level. By Lemma 6.4.10, for any vertex v at depth $\log_{1/2-\epsilon}(1/(2n\epsilon))$ or above,

$$\begin{aligned} n_T(v) &\geq (1/2-\epsilon)^{\log_{1/2-\epsilon}(1/(2n\epsilon))} n \\ &= 1/2\epsilon. \end{aligned}$$

. By the definition of ϵ-relative balance, this means that the balance guarantee holds for the split at this vertex. Furthermore, all internal vertices in the resulting tree T' are at, or above, this level. Thus, all internal vertices except those with leaf children exhibit the relative balance guarantee.

In order to bound the proportional increase in cost, we must bound the operation cost of the level abstraction. Observe that the minimum depth in the abstraction is $\log_{1/2-\epsilon}(1/(2n\epsilon))$. Applying Lemma 6.4.10, we have that the maximum cluster size is at most $e^{4/(c(1-o(1)))}/(2\epsilon) = \frac{1}{2}ce^{4/(c(1-o(1))} \log_2 n$. Since the smallest cluster size involved is at least 1, we can then bound the operation cost by this max cluster size, giving our result. □

Lemma 6.6.8. *Let* T *be an* ϵ*-relatively balanced tree, except for the final layer. After applying tree folding in* FairHC*, any subtree rooted at a child of the root remains* ϵ*-relatively balanced.*

Proof of Lemma 6.6.8. First, note that tree folding only involves overlaying the topology of isomorphic trees (ignoring their leaves). Consider a non-root vertex v in FairHC after tree folding that is not a parent of leaves. As a result of merging the k vertices, $\{v_i\}_{i=1}^k$, we have that v's left and right children are further the result of merging $\{l_i\}_{i=1}^k$ and $\{r_i\}_{i=1}^k$ respectively. Therefore,

$$n_{T'}(v) = \sum_{i\in[k]} n_T(v_i)$$
$$\Rightarrow n_{T'}(l) = \sum_{i\in[k]} n_T(l_i)$$
$$\Rightarrow n_{T'}(r) = \sum_{i\in[k]} n_T(r_i).$$

By the relative balance property, for any $i \in [k]$:

$$(1/2-\epsilon)n_T(v_i) \leq n_T(l_i), n_T(r_i) \leq (1/2+\epsilon)n_T(v_i)$$

Summing over all $i \in [k]$, we obtain

$$(1/2-\epsilon)n_T(v) \leq n_T(l), n_T(r) \leq (1/2+\epsilon)n_T(v)$$

Therefore, the split from v to l and r is relatively balanced. Applying this logic to all such splits, we obtain the result. □

Proof of Lemma 6.4.15. By Lemma 6.4.10, for any vertex v at depth $i \geq \log_2 h$,

$$n_T(v) \geq (1/2-\epsilon)^{\log_2 h} n.$$

For $\epsilon = 1/(c \log n)$ and $h \leq n$, this simplifies to

$$\begin{aligned} n_T(v) &\geq \frac{1}{2^{\log_2 h}} \left(1 - \frac{2}{c \log_2 n}\right)^{\log_2 h} n \\ &\geq e^{-2/c} n/h \end{aligned}$$

For the given depth bounds, $n_T(u)$ for any u is at most n. Thus the level abstraction operation cost is at most $n/(e^{-2/c}n/h) = e^{2/c}h$. □

Proof of Lemma 6.4.16. The operation acting on k tree is by definition. To prove the operation cost, consider internal nodes u and v, in trees T_i and T_j respectively, where $\phi_i(u) = \phi_j(v)$. The isomorphic property of the mapping implies that u and v are at the same height in T_i and T_j respectively. Therefore, they had the same height in the original tree T as well, since the roots of T_i and T_j are both at height $\log_2(h)$ in T. Since both nodes are at height $i \leq \log_{1/2-\epsilon}(1/(2n\epsilon))$, Lemma 6.4.10 yields

$$\frac{n_T(u)}{n_T(v)} \leq e^{4/(c(1-o(1))}$$

This holds for all such pairs u and v, bounding the tree folding operation cost. Note that the ϵ-relative balance property holds after this operation by Lemma 6.6.8. Thus, this argument holds across all tree folds in the for loop. □

Proof of Lemma 6.4.17. Suppose an edge $e = (u, v)$ is separated by the level abstraction operation. This implies that the lowest common ancestor $u \wedge v$ is located at a depth greater than $\log_2 h$. Since the algorithm recurses on clusters at depth $\log_2 h$, in any subsequent recursive instance, the subtree containing e will no longer be included in the recursion. Consequently, e cannot be separated again in any future step of the algorithm. Thus, each edge can be separated by at most

one level abstraction operation.

By assumption, the depth of the last internal node is $\log_{1/2-\epsilon}(1/(2n\epsilon))$. Since we start at subtrees of depth h in the previous dendrogram, we must reduce the depth by $\log_2 h$ at each recursive step. Therefore, there are at most $\log_{1/2-\epsilon}(1/(2n\epsilon))/\log_2 h$ levels of recursion. Leveraging Lemma 6.4.10, we further bound the recursive depth by $\log_2(n)/\log_2(h)$.

At each level of recursion, since e is contained in only one tree, it is only separated by λ tree folds. Therefore, e can only be separated by at most $\lambda \log_2(n)/\log_2(h)$ tree folds. □

Lemma 6.6.9. *For an ϵ-relatively balanced hierarchy T, Algorithm 10 outputs a tree T' such that:*

$$\text{cost}(T') \leq e^{\frac{4\lambda \log_2(n)}{c(1-o(1))\log_2 h}+\frac{2}{c}} \cdot h\text{cost}(T)$$

Proof of Lemma 6.6.9. Lemmas 6.4.15, 6.4.16, and 6.4.17 verify that any edge e can be involved in at most 1 level abstraction of operation cost at most $e^{2/c}h$ and $\lambda \log_2(n)/\log_2(h)$ tree folds of operation cost at most $e^{4/(c(1-o(1))}$ on k trees. By Lemmas 6.6.3 and 6.6.4, this will incur a total proportional cost increase of:

$$\frac{\text{cost}_{T'}(e)}{\text{cost}_T(e)} \leq (e^{4/(c(1-o(1))})^{\lambda \log_2(n)/\log_2(h)} \cdot e^{2/c}h$$

Summing over all edges gives the result. □

Lemma 6.6.10. *Let T be an ϵ-relatively balanced hierarchy with $\nu(V,\ell) = c_\ell n = O(n)$ vertices of each color $\ell \in [\lambda]$. Before recursion,* FairHC *ensures that, for each depth-1 internal node v in*

the output tree T', the fraction of vertices of color ℓ within the subtree of v satisfies

$$\frac{c_\ell}{e^{6/c}} \leq \frac{\ell(v)}{\text{leaves}(v)} \leq c_\ell \left(\frac{e^{4/c}}{kc_\ell} + e^{6/c} \right)$$

for all $\ell \in [\lambda]$.

Proof of Lemma 6.6.10. We start by looking at a single tree fold operator. Assume without loss of generality that we are sorting by the color red. Let $\{v_i\}_{i \in [h]}$ be the resultant ordering from Algorithm 10, r_i be the number of red points from $\text{leaves}(v_i)$, and R be the total number of red vertices.

Fix some i and let v'_i be the root vertex of the resulting subtree in the i-th fold (i.e., the one all the subtrees are mapped onto). By definition, we have the vertices $v_{i+(j-1)k}$ for all $j \in [h/k]$ were folded. Due to the ordering, we know that:

$$r_{i+(j-1)k}/n_T(v_{i+(j-1)k}) \leq r_{y+(j-2)k}/n_T(v_{y+(j-2)k}), \tag{1}$$

$$r_{i+(j-1)k}/n_T(v_{i+(j-1)k}) \geq r_{y+jk}/n_T(v_{y+(j-2)k}) \tag{2}$$

for all $y \in [h/k]$ assuming $j > 1$ for (1) and $j < k$ for (2). Let h' be the (equivalent) height for these three vertices prior to the algorithm's execution. By Lemma 6.4.10:

$$n_T(v_{i+(j-1)k})/n_T(v_{y+(j-2)k}) \leq \frac{(1+2\epsilon)^{\log_2 n}}{(1-2\epsilon)^{\log_2 n}},$$
$$n_T(v_{i+(j-1)k})/n_T(v_{y+jk}) \geq \frac{(1-2\epsilon)^{\log_2 n}}{(1+2\epsilon)^{\log_2 n}}$$

Combining these with the previous inequalities yield:

$$r_{i+(j-1)k} \leq \frac{(1+2\epsilon)^{\log_2 n}}{(1-2\epsilon)^{\log_2 n}} r_{y+(j-2)k},$$
$$r_{i+(j-1)k} \geq \frac{(1-2\epsilon)^{\log_2 n}}{(1+2\epsilon)^{\log_2 n}} r_{y+jk}$$

for all $y \in [h/k]$. Since $\epsilon = 1/(c \log_2 n)$, we further simplify the relation to

$$e^{-4/c} r_{y+jk} \leq r_{i+(j-1)k} \leq e^{4/c} r_{y+(j-2)k}.$$

Since these hold for all y, must further have that

$$\frac{k}{h} e^{-4/c} \sum_{y \in [h/k]} r_{y+jk} \leq r_{i+(j-1)k} \leq \frac{k}{h} e^{4/c} \sum_{y \in [h/k]} r_{y+(j-2)k}.$$

Intuitively, we are partitioning the vertices (in order) into contiguouus chunks of size h/k. Thus, $v_{i+(j-1)k}$ is the i-th vertex in the j-th chunk, and has a lower of fraction of red points than clusters in the previous ($(j-2)$-th) chunk, as well as a higher fraction than clusters in the next (j-th) chunk. If we now let R_{j-1} be the number of reds in the j-th chunk (i.e., $R_{j-1} = \sum_{y \in [h/k]} r_{y+(j-1)k}$), we can further see that $\sum_{j \in [k]} R_{j-1} = R$.

Adopting this notation, we can combine our two previous results such that, for fixed i:

$$\begin{aligned}
\sum_{j\in[k]} r_{i+(j-1)k} &\leq r_i + \frac{k}{h}e^{4/c}\sum_{j\in[k]} R_{j-1} \\
&= r_i + \frac{k}{h}e^{4/c}R, \\
\sum_{j\in[k]} r_{i+(j-1)k} &\geq r_{h-h/k+i} + \frac{k}{h}e^{-4/c}\sum_{j\in[k]} R_{j-1} \\
&= \frac{k}{h}e^{-4/c}R
\end{aligned}$$

Notice that if everything were perfectly balanced, $\frac{k}{h}R$ is exactly the number of reds we need in $\text{leaves}(v_i')$.

We proceed to bound r_i. In the worst case, this can be an entirely red cluster. Thus, we can only provide a weak bound by the size of the cluster at depth $\log_2 h$ by Lemma 6.4.10.

$$\begin{aligned}
r_i &\leq n_T(v_i) \\
&\leq (1/2+\epsilon)^{\log_2 h} n \\
&= 2^{-\log_2 h}(1+2\epsilon)^{\log_2 h} n \\
&\leq e^{2/c} n/h
\end{aligned}$$

where the final inequality comes from the fact that $h \leq n$ and $\epsilon = 1/(c\log n)$. Substituting $R = c_R n$ for some $c_R = O(1)$, we obtain

$$r_i \leq e^{2/c} R/(c_R h).$$

Lemma 6.4.10 gives us that $n_{T'}(v'_i) \geq k(1-2\epsilon)^{\log_2 h} n/h \geq ke^{-2/c}n/h$ and $n_{T'}(v'_i) \leq k(1+2\epsilon)^{\log_2 h} n/h \leq ke^{2/c}n/h$ (applying the same logic as the upper bound to $n_T(v_i)$, k times). Therefore,

$$\begin{aligned}
\frac{\sum_{j\in[k]} r_{i+jk}}{n_{T'}(v'_i)} &\leq \frac{e^{2/c}R/(c_R h) + \frac{k}{h}e^{4/c}R}{ke^{-2/c}(n/h)} \\
&= \frac{R}{n} \cdot \left(\frac{e^{4/c}/c_R}{k} + e^{6/c} \right), \\
\frac{\sum_{j\in[k]} r_{i+jk}}{n_{T'}(v'_i)} &\geq \frac{\frac{k}{h}e^{-4/c}R}{ke^{2/c}(n/h)} \\
&= \frac{R}{n} \cdot \frac{1}{e^{6/c}}
\end{aligned}$$

which bounds the fraction of red points in the cluster. This completes the proof for one tree fold under the observation that $\frac{R}{n} = c_\ell$ if red is ℓ. The same (if not stronger) bounds hold for all subsequent λ tree folds for each color. Lastly, observe that this bound remains valid because merging two clusters, each satisfying the same upper bound on the fraction of red points, preserves the bound. □

Proof of Lemma 6.4.18. By nature of the algorithm, the most imbalanced clusters in this process will be the clusters in the final level of the hierarchy. By Lemma 6.6.10, when we recurse, we have at most an

$$\frac{c_\ell}{e^{6/c}} \leq \frac{\ell(v)}{\text{leaves}(v)} \leq c_\ell \cdot (e^{4/c}/(kc_\ell) + e^{6/c})$$

fraction of vertices of color $\ell \in [\lambda]$. After the at most $\log_2 n / \log_2 h = \log_h n$ recursive levels

guaranteed by Lemma 6.4.17, our bound becomes:

$$\frac{c_\ell}{e^{6t\log_h n/c}} \le \frac{\ell(v)}{\text{leaves}(v)} \le c_\ell \cdot (e^{4/c}/(kc_\ell) + e^{6/c})^{\log_h n}$$

completing the proof. □

We lastly prove the main theorem by combining the lemmas.

Proof of Theorem 6.4.14. Let T^* be the optimal tree, let T_1 be our input tree which is an

$$ce^{4/(c(1-o(1))}\log_2 n \cdot \frac{9\gamma}{4\epsilon}$$

approximation guaranteed by Theorem 6.4.9, but using $t = \log_{1/2-\epsilon}(1/(2n\epsilon))$ (this was shown more explicitly in Lemma 6.6.7), and let T' be our output. By Lemma 6.4.18, T' satisfies our fairness constraints. By Lemmas 6.4.15, 6.4.16, and 6.4.17, every edge is separated by at most 1 level abstraction of max operation cost $e^{2/c}n/h$ and $\log_2(n)/\log_2(h)$ tree folds of operation cost at most $e^{4/(c(1-o(1))}$ on k subtrees. Lemma 6.6.9 immediately tells us:

$$\text{cost}(T') \le e^{\frac{4\lambda\log_2 n}{c(1-o(1))\log_2 h}+\frac{2}{c}} \cdot h\text{cost}(T_1)$$

Combining this with the approximation guaranteed by T_1, we have:

$$\text{cost}(T') \le e^{\frac{4\lambda\log_2 n}{c(1-o(1))\log_2 h}+\frac{2}{c}} \cdot hce^{4/(c(1-o(1))}\log_2 n \cdot \frac{9\gamma}{4\epsilon}\text{cost}(T^*)$$

Simplifying and plugging in $h = n^\delta, \epsilon = 1/(c\log_2 n)$ yields the desired result. □

6.6.3 Algorithmic Runtime Guarantees

Here we analyze the runtime of our four algorithms. Recall that before each of these algorithms, we run a black-box cost-approximate hierarchical clustering algorithm as well as all previous algorithms. For simplicity, here we will present the contribution of each algorithm to the runtime.

Theorem 6.4.1: The algorithm begins at the root and traverses down one side of the tree until it identifies a cluster of a certain minimum size. Once this cluster is found, a tree rebalance operation is applied. The process then recurses on each child subtree.

The traversal depth in each step is at most $O(n)$, and a single tree rebalance operation consists of constant-time pointer adjustments, making its individual runtime $O(1)$. Since this process is initiated from each vertex in the tree, the total runtime is given by $O(n^2)$.

Theorem 6.4.5: Similar to the previous algorithm, this algorithm performs computations at each of the $O(n)$ nodes in the tree. At each node, it executes a subtree search operation until the desired balance condition is met. Given that the current balance δ is reduced to at most $\frac{2\delta}{3}$ at each step, the total number of steps required to achieve an ϵ-relative balance is $O\left(\log \frac{1}{\epsilon}\right)$. Each subtree search operation consists of two traversals of length $O(n)$ to locate the appropriate insertion and deletion points. Apart from these traversals, all other operations involve constant-time pointer manipulations. Therefore, the overall runtime of the algorithm is $O\left(n^2 \log \frac{1}{\epsilon}\right)$.

Theorem 6.4.9: This algorithm only deletes a set of low nodes in the tree. Thus it only requires $O(n)$ time.

Theorem 6.4.14: As in previous steps, the algorithm performs computations on at most $O(n)$ nodes in the tree. Each tree abstraction step requires $O(n)$ time, following the same reasoning as

before.

Additionally, computing the fraction of red and blue vertices in each considered cluster and subsequently sorting them requires $O(n)$ time. The final step, where vertices are folded on top of each other, involves identifying an isomorphism between subtrees. This is achieved by indexing the vertices within each subtree and directly applying the isomorphism, both of which can be performed in $O(n)$ time. Thus, the total runtime for this stage is $O(n^2)$. Therefore, the total runtime of the final algorithm (excluding the black-box step) is bounded by the computation time from Theorem 6.4.5, which is $O\left(n^2 \log \frac{1}{\epsilon}\right)$. Since ϵ is lower-bounded by $\epsilon > 1/n$, this simplifies to $O(n^2 \log n)$.

6.7 Conclusions, Limitations & Future Work

The main limitations this work suffers from encapsulate the general limitations of study in theoretical clustering fairness. Our work strives to provide algorithms that are applicable to many hierarchical clustering applications where fairness is a concern. However, our work is inherently limited by its focus on a specific fairness constraint (i.e., the extension of disparate impact originally used to study fair clustering [Chierichetti et al., 2017]). While disparate impact has received substantive attention in the clustering community and is seen as one of the primary fairness definitions/constraints [Ahmadian et al., 2020a, Ahmadian et al., 2020b, Bera et al., 2019, Brubach et al., 2020, Kleindessner et al., 2019b], it is just one of many established fairness constraints for problems in clustering [Chakrabarti et al., 2022, Esmaeili et al., 2021, Kleindessner et al., 2019a]. When applying fair machine learning algorithms to problems, it is not always clear which fairness constraints are the best for the application. This, and the fact that the application of fairness

to a problem can cause harm in other ways [Ben-Porat et al., 2021], means that the proposal of theoretical fair machine learning algorithms always has the potential for improper or even harmful use. While this work proposes purely theoretical advances to the field, we direct the reader to [Barocas et al., 2019] for a broader view on the field.

Our results are also limited by the theoretical assumptions that we make. For instance, in the stochastic fairness algorithm, we assume that the probabilities of a vertex being a certain color are within the same bounds across all vertices. This may not be realistic, as there could be higher variance in the distribution of color probabilities, and even though the probabilities may lie outside of our assumed bounds, it still may be tractable to find a low-cost hierarchical clustering.

In our main theorem, we assume that there are only two colors (protected classes), and that they subsume a constant fraction of the general population. The former assumption is clearly limited in that in many cases, protected classes may take on more than two values. The constant fraction assumption is actually highly relevant and is reflected in other clustering literature, but it is a potential limitation that may rule out a handful of applications nevertheless.

Finally, our results are limited to the evaluation of hierarchical clustering quality based off cost. While this is a highly regarded metric for hierarchy evaluation, there may be situations where others are appropriate. It also neglects the practicality of empirical study in that many important machine learning algorithms we use today cannot provide guarantees across all data (which our results necessarily do), but they perform much better on most actual inputs. However, we leave it as an open question to further evaluate the practicality of our algorithms through empirical study.

Chapter 7: Fair Polylog Approximate Hierarchical Clustering

7.1 Introduction

Clustering is a pervasive machine learning technique which has filled a vital niche in every day computer systems. Extending upon this, a *hierarchical clustering* is a recursively defined clustering where each cluster is partitioned into two or more clusters, and so on. This adds structure to flat clustering, giving an algorithm the ability to depict data similarity at different resolutions as well as an ancestral relationship between data points, as in the phylogenetic tree [Kraskov et al., 2003].

On top of computational biology, hierarchical clustering has found various uses across computer imaging [Chen et al., 2020a, Selvan et al., 2005], computer security [Chen et al., 2020b, Chen et al., 2020a], natural language processing [Ramanath et al., 2013], and much more. Moreover, it can be applied to any flat clustering problem where the number of desired clusters is not given. Specifically, a hierarchical clustering can be viewed as a structure of clusterings at different resolutions that all agree with each other (i.e., two points clustered together in a higher resolution clustering will also be together in a lower resolution clustering). Generally, hierarchical clustering techniques are quite impactful on modern technology, and it is important to guarantee they are both effective and unharmful.

Researchers have increasingly recognized the harmful biases that arise in unchecked ma-

chine learning systems. These biases have led to documented cases of racial discrimination across various domains, including the allocation of healthcare resources [Ledford, 2019], the presentation of arrest-related advertisements [Sweeney, 2013], hiring predictions [Bogen and Rieke, 2018], and recidivism risk assessments [Angwin et al., 2016b]. A widely studied approach to addressing such biases is fair machine learning, which aims to mitigate discrimination, particularly against protected groups. In this framework, fairness is often formalized as a constraint imposed on the solution space, ensuring that optimization is performed while adhering to specific fairness criteria. One influential fairness concept, individual fairness, was introduced by [Dwork et al., 2012]. This principle requires that similar individuals receive similar treatment in classification or decision-making processes, providing a formal mechanism for reducing bias in machine learning models.

In line with previous work in clustering and hierarchical clustering, this chapter utilizes the notion of *group fairness*, which enforces that different protected classes receive a proportionally similar distribution of classifications (in our case, cluster placement). [Chierichetti et al., 2017] first introduced this as a constraint for the flat clustering problem, arguing that it mitigates a system's disparate impact, or non-proportional impact on different demographics. This notion of fair clustering has been similarly leveraged and extended by a vast range of works in both flat [Ahmadian et al., 2019, Bera et al., 2019, Bercea et al., 2019] and hierarchical [Ahmadian et al., 2020a, Knittel et al., 2023b] clustering.

To illustrate our fairness concept, consider the following application (Figure 7.1): a news database is structured as a hierarchical clustering of search terms, where a search term is associated with a cluster of news articles to output to the reader, and more specific search terms access finer-resolution clusters. When a user searches for a term, we simply identify the corre-

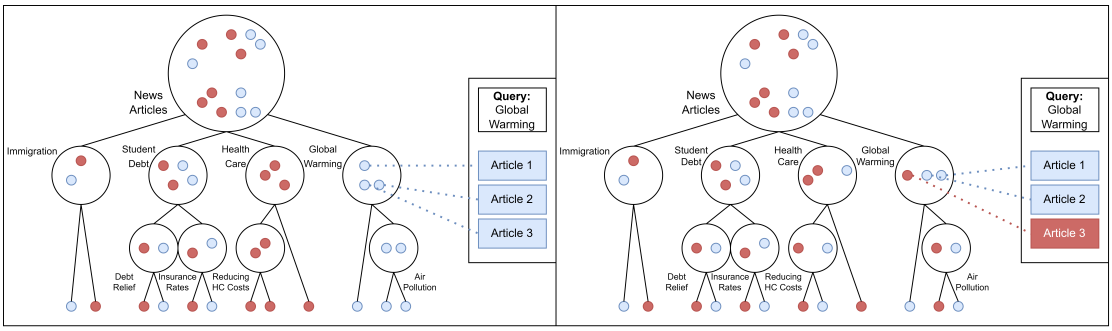


Figure 7.1: A hierarchical clustering of news articles. Red articles are conservative, blue are liberal. On the left is the optimal unfair hierarchy. We alter the hierarchy slightly on the right to achieve fairness. Now, the user's query for global warming will yield both liberal and conservative articles.

sponding cluster and output the contained articles. However, as is, the system does not account for the political skew of articles. In Figure 7.1, we label conservative-leaning articles in red and liberal-leaning articles in blue. We can see that in this example, when the user searches for global warming articles, they will only see liberal articles. To resolve this, we add a group fairness constraint on our cluster: for example, require at least 1/3 of the articles in each cluster to be of each political skew. This guarantees (as depicted on the right) that the outputted articles will always be at least 1/3 liberal and 1/3 conservative, thus guaranteeing the user is exposed to a range of perspectives along this political axis. This notion of fairness, which we formally define in Definition 7.2.3, has been explored in the context of hierarchical clustering in both [Ahmadian et al., 2020a] and [Knittel et al., 2023b].

This chapter focuses on approximation algorithms for fair *low-cost* hierarchical clustering. The cost metric, first introduced by [Dasgupta, 2016] and formally defined in Definition 7.2.2, is one of the most natural and well-motivated measures for evaluating hierarchical clustering solutions. Intuitively, each pair of points contributes an additive cost of similarity $\times$ clust_size, where clust_size is the size of the smallest cluster in the hierarchy that contains both points.

Optimizing this cost metric is challenging. The best-known approximation for the standard (unfair) problem is an $O(\sqrt{\log n})$-approximation [Charikar and Chatziafratis, 2017, Dasgupta, 2016]. Moreover, it is conjectured that no constant-factor ($O(1)$) approximation exists [Charikar and Chatziafratis, 2017]. Incorporating fairness constraints makes the problem even more difficult. The first work to address fair low-cost hierarchical clustering, [Ahmadian et al., 2020a], achieved an impractical $O(n^{5/6}\log^{3/2} n)$-approximation—only marginally better than the trivial $O(n)$ upper bound. This highlighted the inherent complexity of the problem. A major improvement was later made by [Knittel et al., 2023b], who achieved a near-polylogarithmic approximation of $O(n^{\delta}\text{polylog}(n))$, where δ can be made arbitrarily small, allowing for a trade-off between fairness and approximation quality.

Despite this progress, obtaining a true polylogarithmic approximation remained an open problem. In this chapter, we resolve this by presenting the first polylogarithmic approximation for fair low-cost hierarchical clustering.

7.1.1 Our Contributions

This work proposes the first polylogarithmic approximation for fair, low-cost hierarchical clusterings. We build on the work of [Knittel et al., 2023b] as a starting point and develop a much simpler, more direct algorithm that achieves both improved fairness and better approximation guarantees. Our procedure, as in the prior chapter, starts with a low-cost unfair hierarchical clustering and then alters it with multiple well-defined and limited tree operators. This gives it a degree of explainability, in that the user can understand exactly the steps the algorithm took to achieve its result and why. In addition, our algorithm achieves both relative cluster balance

(i.e., clusters who are children of the same cluster have similar size) and fairness, along with a parameterizeable trade-off between the two.

On top of the benefits of [Knittel et al., 2023b]'s techniques, we propose a greatly simplified algorithm. Specifically, the initially proposed algorithm required four tree operators, however, we only require two of the four. This makes the algorithm simpler to understand and more implementable. We show that even with this reduced functionality, we can cleverly achieve both a better approximation and degree of fairness:

Theorem 7.1.1 (High-Level)**.** *Given a γ-approximation to the cost objective for a hierarchical clustering on a set of n points, Algorithm 2 yields an $O(\log^2 n)$ approximation to cost which is relatively fair in time $O(n \log^2 n)$. Moreover, all clusters are $O(1/\log n)$ balanced, i.e., any cluster's children clusters are within $O(1/\log n)$ size different.*

To put this in perspective, previously, the best approximation for fair hierarchical clustering previously was $O(n^{\delta}\text{polylog}(n))$, whereas the best unfair approximation is $O(\sqrt{\log n})$. Our work greatly reduces this gap by providing a true $O(\text{polylog}(n))$ approximation, with a low degree of 2. Note that this is not the most general and rigorous form of our result – for that, we defer to Theorem 7.3 as stated in the body of our work.

7.2 Preliminaries

7.2.1 The Vanilla Problem

Fair clustering literature refers to the original problem variant, without fairness, as the "vanilla" problem. We define the vanilla problem of finding a low-cost hierarchical clustering

here using our specific notation.

We are given a complete graph $G = (V, E, w)$ with a weight function $w : E \to \mathbb{R}^+$ as a measure of the similarity between datapoints. Note the data is encoded as a complete tree because we require knowledge of all point-point relationships. We must construct a hierarchical clustering, represented by its dendrogram, T, with root denoted $\text{root}(T)$. T is a tree with vertices corresponding to all clusters of the hierarchical clustering. Leaves of T, denoted $\text{leaves}(\text{root}(T))$ correspond to the points in the dataset (i.e., singleton clusters). An internal node v corresponds to the cluster containing all leaf-data of the maximal subtree (i.e., contains all its descendants) rooted at v, $T[v]$. In addition, we let $u \wedge v$ denote the lowest common ancestor of u and v in T.

In order to define [Dasgupta, 2016]'s cost function, we use the same notational simplifications as [Knittel et al., 2023b]. For an edge $e = (x, y) \in E$, we say $n_T(e) = |\text{leaves}(T[x \wedge y])|$ is the size of the smallest cluster in the hierarchy containing e. Similarly, for a hierarchy node v, $n_T(v_i) = |\text{leaves}(T[v_i])|$ is the size of the corresponding cluster. This is sufficient to introduce the notion of *cost*.

Definition 7.2.1 ([Knittel et al., 2023b]). The **cost** of $e \in E$ in a graph $G = (V, E, w)$ in a hierarchy T is $\text{cost}_T(e) = w(e) \cdot n_T(e)$.

Dasgupta's cost function can then be written as a sum over the costs of all edges.

Definition 7.2.2 ([Dasgupta, 2016]). The **cost** of a hierarchy T on graph $G = (V, E, w)$ is:

$$\text{cost}(T) = \sum_{e \in E} \text{cost}_T(e)$$

Our algorithm begins by assuming we have some approximate vanilla hierarchy, T. That

is, if OPT is the optimal hierarchy tree, then $\text{cost}(T) \leq \alpha \cdot \text{cost}(OPT)$ for some approximation factor α. According to [Dasgupta, 2016], we can transform this hierarchy to be binary without increasing the cost, thus we here assumes our input is binary. We then produce a modified hierarchy T' which similar structure to T that guarantees fairness, i.e., $\text{cost}(T') \leq \alpha' \cdot \text{cost}(OPT)$ for some approximation factor $\alpha' \geq \alpha$. Note that the binary assumption may not hold when we consider adding a fairness constraint.

We emphasize that this comparison is being made to the *vanilla* OPT. It is currently unclear how to classify the optimal fair hierarchy, and this problem remains an interesting direction of future research.

7.2.2 Fairness and Balance Constraints

We consider the fairness constraints based off those introduced by [Chierichetti et al., 2017] and extended by [Bercea et al., 2019]. On a graph G with colored vertices, let $\nu(C, \ell)$ count the number of ℓ-colored points in cluster C.

Definition 7.2.3 ([Knittel et al., 2023b])**.** Consider a graph $G = (V, E, w)$ with vertices colored one of λ colors, and two vectors of parameters $\alpha, \beta \in (0, 1)^\lambda$ with $\alpha_\ell \leq \beta_\ell$ for all $\ell \in [\lambda]$. A hierarchy T on G is **fair** if for any non-singleton cluster C in T and for every $\ell \in [\lambda]$, $\alpha_\ell |C| \leq \nu(C, \ell) \leq \beta_\ell |C|$. Additionally, any cluster with a leaf child has only leaf children.

Effectively, we are given bounds α_ℓ and β_ℓ for each color ℓ. Every non-singleton cluster must then have at least an α_ℓ fraction and at most a β_ℓ fraction of color ℓ. This guarantees proportional representational fairness of each color in each cluster.

As an intermediate step in achieving fairness, we will create splits in our hierarchy that

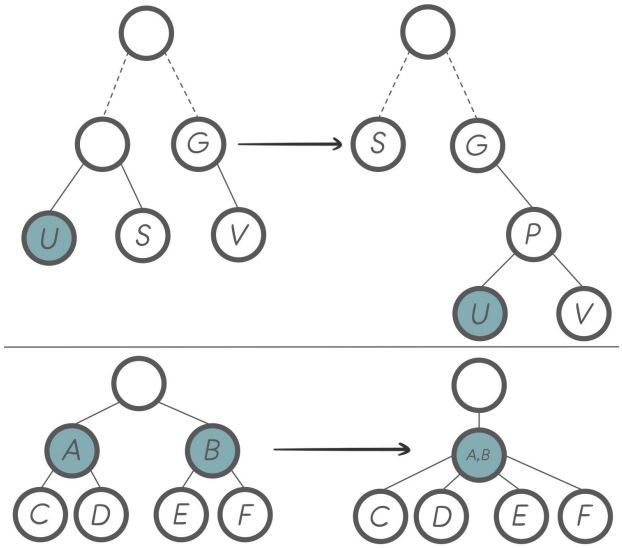


Figure 7.2: Our operators: subtree deletion and insertion (upper panel) and shallow tree folding (lower panel).

achieve relative balance in terms of subcluster size. Thus, we will consistently characterize trees with the following definition.

Definition 7.2.4. In a hierarchy, a vertex v (corresponding to cluster C) with c_v children is **ϵ-relatively balanced** if for every cluster $\{C_i\}_{i\in[c_v]}$ that corresponds to a child of v, $(\frac{1}{c_v} - \epsilon)|C| \leq |C_i| \leq (\frac{1}{c_v} + \epsilon)|C|$.

While this definition is quite similar to that from [Knittel et al., 2023b] (see Chapter 7), it deviates in two ways: 1) we only define it on a single split as opposed to the entire hierarchy and 2) we allow splits to be non-binary. If we apply it to the entire hierarchy and constrain it to be binary, it is equivalent to the former definition.

7.2.3 Tree Operators

Our work simplifies the work of [Knittel et al., 2023b]. In doing so, we follow the same framework, using tree operators to make well-defined and limited alterations to a given hierar-

chical clustering (Figure 7.2). In addition, our algorithm simplifies operator use in two ways: 1) we only utilize two of their four tree operators, and 2) we greatly simplified the complicated "tree-folding" operator, showing that it can still be used to create a fair hierarchy.

We first review the subtree deletion and insertion operator, which we do not modify from the prior algorithm. However, the manner in which we apply this operation is modified to achieve better guarantees (see Section 7.3). At a high level, this operator removes a subtree from one part of the hierarchy and reinserts it elsewhere, adding and removing parent vertices as necessary.

Definition 7.2.5 ([Knittel et al., 2023b]). Consider a binary tree T with internal nodes u, some non-ancestor v, u's sibling s, and v's parent g. **Subtree deletion** at u removes $T[u]$ from T and contracts s into its parent. **Subtree insertion** of $T[u]$ at v inserts a new parent p between v and g and adds u as a second child of p. The operator $\mathrm{del\_ins}(u, v)$ deletes u and inserts $T[u]$ at v.

The more complicated operation which we here simplify substantially is the tree folding. In the previous work, this operation took two or more isomorphic trees and mapped the internal nodes to each other. We here instead simply take two or more subtrees and merge their roots. We henceforth refer to our modified operator as the "shallow tree fold."

Definition 7.2.6. Consider a set of subtrees $T_1, \ldots, T_k$ of T such that all $\mathrm{root}(T_i)$ have the same parent p in T. A **shallow tree folding** of trees $T_1, \ldots, T_k$ ($\mathrm{fold}(T_1, \ldots, T_k)$) modifies T such that all $T_1, \ldots, T_k$ are replaced by a single tree T_f whose root $\mathrm{root}(T_f)$ is made a child of p, and whose root-children are $\cup_{i \in [k]} \mathrm{children}(\mathrm{root}(T_i))$.

While making this modification to the hierarchy breaks the binary guarantee, we can arbitrarily binarize the subtree after folding without impacting the cost guarantees [Dasgupta, 2016]. Since our algorithm works top-bottom, creating balanced vertices as it goes, we don't yet care

about the fairness of the descendants of T_f. Following the rebalancing steps, we will then recursively call our algorithm on on the folded subtrees precisely to rectify the fairness violations.

7.3 Main Algorithm

In this section, we present our fair, low-cost, hierarchical clustering algorithm along with the crucial lemmas that combine to give the final guarantees. At each step, we give intuition for the the theoretical results, and prove their correctness in Section 7.5.

Ultimately, we achieve the following: When T is a γ-approximate low-cost vanilla hierarchical clustering over $\ell(V) = c_\ell n = O(n)$ vertices of each color $\ell \in [\lambda]$, MakeFair (Algorithm 13), for any constants ϵ, h, k with $h \gg k^\lambda$ and $n \gg h$, runs in $O(n \log n(h + \lambda \log n))$ time and yields a hierarchy T' satisfying:

1. T' is an $O\left(\frac{(h-1)}{\epsilon} + \frac{1+\epsilon}{1-\epsilon}k^\lambda\right)\gamma$-approximation for cost.
2. T' is fair for any parameters for all $i \in [\lambda]$: $\alpha_i \leq \frac{\lambda_i}{n}\left(\frac{1-\epsilon}{(1+\epsilon)^2}\left(1 - \frac{k(1+\epsilon)}{c_i h}\right)\right)^{O(\log(n))}$ and $\beta_i \geq \frac{\lambda_i}{n}\left(\frac{1+\epsilon}{(1-\epsilon)^2}\left(1 + \frac{1-\epsilon}{c_i k}\right)\right)^{O(\log(n))}$, where $\lambda_i = c_i n$.
3. All internal nodes in T' are ϵ-relatively balanced.

The simpler version that was presented in Section 7.1 is obtained by setting $k = O(1)$, $h = O(\log n)$, and $\epsilon = O(1/\log n)$. Further note that we assume $\lambda = O(1)$.

The core idea of our algorithm is to leverage similar tree operators as those used in [Knittel et al., 2023b], but to significantly streamline their application, making them both more direct and carefully controlled. In contrast to previous work, which processes the tree in four separate stages—first to enforce $\frac{1}{6}$-relative balance, then to refine it to ϵ-relative balance, followed by removing the bottom layers of the hierarchy, and finally applying fairness constraints—our ap-

proach eliminates the inefficiencies introduced by these sequential modifications. The key issue with the prior method is that each stage compounds proportional cost increases, leading to an exponential growth in cost distortion, particularly due to the degradation of relative balance as one moves deeper in the hierarchy.

To address this, we instead perform a single top-to-bottom pass over the tree, simultaneously rebalancing and folding clusters to enforce fairness as we progress. This streamlined approach avoids unnecessary recomputation and prevents the accumulation of excessive cost distortions. We now describe the algorithm in detail.

Our approach takes as input a given hierarchical clustering tree. While starting from a tree with a good approximation to the vanilla problem will yield a better final clustering, our results work as a black box on top of *any* initial hierarchical clustering. We first apply $\mathrm{SplitRoot}$ in order to balance the root (Section 7.3.1). Subsequently, we leverage shallow tree folding on the children of the root to achieve fairness (Section 7.3.2). This gives us the first layer of our output, and then we recurse down the tree.

7.3.1 Root Splitting and Balancing

Our first subroutine is $\mathrm{SplitRoot}$, depicted in Algorithm 12. At a high level, the procedure rebalances the root, and immediately splits it into h children (where h is a parameter to be tuned). This effectively fills the role of the "Refine Rebalance Tree" from [Knittel et al., 2023b].

We begin $\mathrm{SplitRoot}$ by adding dummy children to v until it has exactly h children, noting that we can assume the input tree is binary. A dummy (or null) child serves as a placeholder for a subtree yet to be constructed and can be thought of as a zero-sized tree.[1] Importantly, by the end

[1] Adding dummy nodes does not increase the number of leaves in the tree.

of the process, all dummy children will be replaced with actual subtrees.

Next, we define v_{max} and v_{min}, the maximal subtrees rooted at $\text{children}(\text{root}(T'))$ which have the most and fewest leaves, respectively. As long as the root is not ϵ-relatively balanced (which is equivalent to $n_{T'}(v_{max})$ or $n_{T'}(v_{min})$ deviating from the target n/h by over $n\epsilon$, as they are extreme points), we will attempt to rebalance. We define δ_1 and δ_2 to be the proportional deviation of $n_{T'}(v_{min})$ and $n_{T'}(v_{max})$ from the target size n/h respectively, and δ to be the minimum of the two. In effect, δ measures the maximum number of leaves we can move from the large subtree to the small subtree without causing $n_{T'}(v_{max})$ to dip below n/h or $n_{T'}(v_{min})$ to peak above n/h. This is important to guarantee our runtime: as an accounting scheme, we show that clusters monotonically approach size n/h, and thus we can quantify how fast our algorithm completes. We fully analyze this later in Lemma 7.3.2.

We now invoke exactly this procedure: move a large subtree from v_{max} to v_{min}, though this subtree can have no more than δn leaves. To efficiently achieve this, we start by visiting v_{max} and traversing down its right (larger) children (recall below v_{max}, the tree is still binary). We halt on the first child that is of size δn or smaller. We then remove it and find a search for a node to reinsert it under v_{min}.

The insertion spot is found similarly by descending down v_{min}'s left (smaller) children until the right child of the current vertex has fewer leaves in its subtree than the tree we are inserting. Thus, we have completed our insertion and deletion operations. We repeat until the tree is relatively balanced as desired.

Algorithm 12 SplitRoot

Require: A binary hierarchy tree T of size $n \geq 1/2\epsilon$ over a graph $G = (V, E, w)$, with smaller cluster always on the left, and parameters $h \in [n]$ and $\epsilon \in (0, \min(1/6, 1/h))$.

Ensure: A hierarchical clustering T' with an ϵ-relatively balanced root that has k children.

1: Initialize $T' = T$
2: $v = \text{root}(T')$
3: Add null children to v until it has h children
4: Let $v_{min} = \text{argmin}_{v' \in \text{children}(v)} n_{T'}(v')$
5: Let $v_{max} = \text{argmax}_{v' \in \text{children}(v)} n_{T'}(v')$
6: **while** $n_{T'}(v_{max}) > n(1/h + \epsilon)$ or $n_{T'}(v_{min}) < n(1/h - \epsilon)$ **do**
7: $\quad \delta_1 = 1/h - n_{T'}(v_{min})/n$
8: $\quad \delta_2 = n_{T'}(v_{max})/n - 1/h$
9: $\quad \delta = \min(\delta_1, \delta_2)$
10: $\quad$ Let $v = v_{max}$
11:
12: $\quad$ **while** $n_{T'}(v) > \delta n$ **do**
13: $\quad\quad v \leftarrow \text{right}_{T'}(v)$
14: $\quad$ **end while**
15:
16: $\quad u \leftarrow v_{min}$
17: $\quad$ **while** $n_{T'}(\text{right}_{T'}(u)) \geq n_{T'}(v)$ **do**
18: $\quad\quad u \leftarrow \text{left}_{T'}(u)$
19: $\quad$ **end while**
20: $\quad T' \leftarrow T'.\text{del\_ins}(u, v)$
21: $\quad$ Reset v_{min} and v_{max}
22: **end while**

We now analyze this subroutine, providing intuition below, with full proofs presented in Section 7.5. To start, consider the tree we are deleting and reinserting, denoted $T'[v]$. To ensure that this subtree has a sufficient number of leaves to move, we demonstrate that:

Lemma 7.3.1. *For a subtree $T'[v]$ that is deleted and reinserted in* SplitRoot *(Algorithm 12), we must have that $\frac{\epsilon n}{2(h-1)} < n_T(v) \leq \delta n$.*

The upper bound comes directly from the stopping condition of the while loop in Line 12. To conclude the lower bound, we start by noting that $\max(\delta_1, \delta_2) > \epsilon$, as otherwise the stopping condition for the outer loop would be met. Consider further the total amount of "excess of large

clusters" or, more precisely, the sum over all deviations from n/h of clusters larger than n/h (note if all clusters were n/h, it would be perfectly balanced). This total excess must be matched in the "deficiency of small clusters", which is the sum of deviations of clusters smaller than n/h. Therefore, since there are at least h small or h large clusters, the largest deviation must be at most h times the smallest deviation, according to our accounting scheme. This allows us to bound $\delta \geq \epsilon/(h-1)$. The tree that is inserted and deleted must have at least half this many leaves, since it is the larger child of a node with over δn leaves in its subtree. This yields the lower bound, showing we move at least a significant number of vertices each step.

We now illustrate the relative balance of the subroutine. In addition to our analysis, we also obtain the runtime, which exhibits near-linearity when the condition $h \ll n$ holds.

Lemma 7.3.2. SplitRoot *(Algorithm 12) yields a hierarchy whose root is ϵ-relatively balanced with h children. In addition, it requires $O(nh)$ time to terminate.*

To see the result, note that the root has h children by definition at the algorithm start and this remains invariant. The runtime follows from our accounting scheme: at each step, the total excess and deficiency decrease by at least the number of leaves in the subtree being moved. By Lemma 7.3.1, this number is at least $\frac{n\epsilon}{2(h-1)}$, ensuring that the process converges in $O(h)$ steps. Each step involves searching for insertion and deletion points, which can be performed in $O(n)$ time. Thus, the overall runtime is bounded by $O(nh)$. Finally, the balance guarantee follows directly from our stopping condition, which ensures that the root is relatively balanced upon termination.

All that remains is to show the negative impact on the cost of edges that are separated by the algorithm. We bound this via the following lemma.

Lemma 7.3.3. *In* SplitRoot *(Algorithm 12), for all* $e \in E$ *that are separated:*

$$\text{cost}_{T'}(e) \leq n \cdot w(e) \leq \frac{2(h-1)}{\epsilon} \cdot \text{cost}_T(e)$$

Lemma 7.3.1 gives us that moved subtrees are of size at least $\frac{\epsilon n}{2(h-1)}$. Note that any edge separated by the routine must have at most one endpoint in the deleted subtree, and one outside. Thus, their least common ancestor is an ancestor of the moved subtree, and must then further have the above lower bound. In the worst case, the final size of the smallest cluster containing such an edge is n, so the proportional increase is at most $\frac{2(h-1)}{\epsilon}$.

7.3.2 Fair Tree Folding

Next, we discuss how to achieve fairness by using MakeFair, as seen in Algorithm 13. This is our final recursive algorithm which utilizes SplitRoot: we start by running the routine to balance the split at the root and give it h children. Next we use a shallow variant on the folding process of [Knittel et al., 2023b] to adhere to the fairness constraints.

More specifically, we first sort the children of the root by the proportional representation of the first color (say, red without loss of generality). Then, we do a shallow fold across various k-sized sets, defined as follows: according to our ordering over the children, partition the vertices into k contiguous chunks starting from the first vertex. For each $i \in [h/k]$, we find the i-th vertex in each chunk and fold them together. Notice that this is a k-wise fold since there are k chunks, and we end up with h/k vertices. We repeat this process for each color, and subsequently recurse on the children. If a child is too small to be balanced by SplitRoot, then we stop and give it a trivial topology (a root with many leaf-children).

This concludes our algorithm description. Next, we analyze its runtime, fairness guarantees, and approximation factor. We begin by examining the degree of fairness achieved at the top level of the hierarchy.

Lemma 7.3.4. MakeFair *(Algorithm 13) yields a hierarchy such that all depth 1 vertices satisfy fairness under* $\alpha_i \leq \frac{\lambda_i}{n} \cdot \frac{1-\epsilon}{(1+\epsilon)^2}\left(1 - \frac{k(1+\epsilon)}{c_i h}\right)$ *and* $\beta_i \geq \frac{\lambda_i}{n} \cdot \frac{1+\epsilon}{(1-\epsilon)^2}\left(1 + \frac{1-\epsilon}{c_i k}\right)$*, where* $\lambda_i = c_i n$*.*

At a high level, we here show that the folding process guarantees a level of fairness. By ordering the parts in our partition by the density of a color, and folding across vertices in each such part, we ultimately distribute the red colored points evenly across our final subtrees. This guarantees a degree of balance.

The problem is that the degree of *fairness* still exhibits a compounding affect as we recurse. More specifically, since the first children are not perfectly balanced, then in the next recursive step, the total data subset we are working on may now deviate from the true color proportions. This deviation is bounded by our result in Lemma 7.3.4, but it will increase proportionally at each step.

Lemma 7.3.5. *In* MakeFair *(Algorithm 13), let* $\{\lambda_i\}_{i\in[\lambda]}$ *be the proportion of each color and assume* $k^\lambda \ll h$*. At any recursive call, the proportion of any color is (where* $\lambda_i = 1/c_i$ *for constant* c_i*):*

$$\lambda_i \left(\frac{1-\epsilon}{(1+\epsilon)^2}\left(1 - \frac{k(1+\epsilon)}{c_i h}\right)\right)^{O(\log(n/h))} \leq \lambda_i^j \leq \lambda_i \left(\frac{1+\epsilon}{(1-\epsilon)^2}\left(1 + \frac{1-\epsilon}{c_i k}\right)\right)^{O(\log(n/h))}$$

where j *is the recursive depth. Moreover, the recursive depth is bounded above by* $O(\log(n/h))$*.*

This result comes directly from Lemma 7.3.4. Effectively, we increase the proportion of

Algorithm 13 MakeFair

Require: A hierarchy tree T of size $n \geq 1/2\epsilon$ over a graph $G = (V, E, w)$ with vertices given one of λ colors, and parameters $h \in [n]$, $k \in [h/(\lambda - 1)]$, and $\epsilon \in (0, \min(1/6, 1/h))$.
Ensure: A fair hierarchical clustering T'.

$T' = \text{SplitRoot}(T, h, \epsilon)$
$h' \leftarrow h$
for each color $\ell \in [\lambda]$ **do**
    Order $\{v_i\}_{i \in [h']} = \text{children}(\text{root}(T'))$ decreasing by $\frac{\ell(\text{leaves}(v_i))}{n_{T'}(v_i)}$
    For all $i \in [k]$, $T' \leftarrow T'.\text{fold}(\{T'[v_{i+(j-1)k}] : j \in [h'/k]\})$
    $h' \leftarrow h'/k$
end for
for each child v_i of $\text{root}(T')$ **do**
    if $n \geq \max(1/2\epsilon, h)$ **then**
        Replace $T'[v_i] \leftarrow \text{MakeFair}(T'[v_i], h, k, \epsilon)$
    else
        Replace $T'[v_i]$ with a tree of root v_i, leaves $\text{leaves}(T'[v_i])$, and depth 1.
    end if
end for

each color by the same factor each recursive step and need only bound the recursive depth. Notice that we start with n vertices and, after splitting, our subtrees have size at most $(1 + \epsilon)n/h$. After one fold, this is increased by a factor of k, and thus k^λ after all folds. Crucially, this does not impact the final result significantly. The procedure, similar to turning an n-sized tree into an n/h-sized tree, yields an $O(\log(n/h))$ recursive depth. This will be sufficient to show our fairness.

Next, we evaluate the cost incurred at each stage in the hierarchy.

Lemma 7.3.6. *In* MakeFair *(Algorithm 13), for all $e \in E$ that is separated before the recursive call:*

$$\text{cost}_{T'}(e) \leq O\left(\frac{2(h-1)}{\epsilon} + \frac{1+\epsilon}{1-\epsilon}k^\lambda\right)\text{cost}_T(e)$$

As discussed before, the final cluster size will be of size $(1 + \epsilon)nk^\lambda/h$. Furthermore, any separated edge must have a starting cluster size of at least $(1 - \epsilon)n/h$, as this is the size of the smallest cluster involved in tree folding. From this, it is straightforward to compute the

proportional cost increase of a single recursive level, as well as the cost increase from the initial splitting in Lemma 7.3.3.

We further demonstrate that, whenever an edge is separated, its endpoints' least common ancestor will no longer be involved in any further recursive step.

Lemma 7.3.7. *In* MakeFair *(Algorithm 13), any edge $e \in E$ is separated at only one level of recursion.*

Putting these two together pretty directly gives us our cost approximation.

Lemma 7.3.8. *In* MakeFair *(Algorithm 13),* $\text{cost}(T') \leq O\left(\frac{2(h-1)}{\epsilon} + \frac{1+\epsilon}{1-\epsilon}k^{\lambda}\right)\text{cost}(T)$.

Finally, Theorem 7.3 comes directly from Lemmas 7.3.6 and 7.3.8.

7.4 Simulations

This section validates the theoretical guarantees of Algorithm 13. Specifically, we demonstrate that modifying an unfair hierarchical clustering using the presented procedure yields a fair hierarchy that incurs only a modest increase in cost.

Datasets. We use two data sets, *Census* and *Bank*, from the UCI data repository [Dua and Graff, 2017]. Within each, we subsample only the features with numerical values. To compute the *cost* of a hierarchical clustering we set the similarity to be $w(i,j) = \frac{1}{1+d(i,j)}$ where $d(i,j)$ is the Euclidean distance between points i and j. We color data based on binary (represented as blue and red) protected features: *race* for *Census* and *marital status* for *Bank* (both in line with the prior work of [Ahmadian et al., 2020a]). As a result, *Census* has a blue to red ratio of 1:7 while *Bank* has 1:3. We then subsample each color in each data set such that we retain (approximately) the data's original balance. We use samples of size 512 for the balance experiments,

and vary the sample sizes when assessing cost. For each experiment we conduct 10 independent replications (with different random seeds for the subsampling), and report the average results. We vary the parameters (h, k, ε) to experimentally assess their theoretical impact on the approximate guarantees of Section 7.3.

Implementation. The Python code for the following experiments are available in the Supplementary Material. We start by running average-linkage, a popular hierarchical clustering algorithm. We then apply Algorithm 13 to modify this structure and induce a *fair* hierarchical clustering that exhibits a mild increase in the cost objective.

Metrics. In our results we track the approximate cost objective increase as follows: Let G be our given graph, T be average-linkage's output, and T' be Algorithm 13's output. We then measure the ratio $\text{RATIO}_{cost} = cost_G(T')/cost_G(T)$. We additionally quantify the fairness that results from application of our algorithm by reporting the balances of each cluster in the final hierarchical clustering, where true fairness would match the color proportions of the underlying dataset.

Results. We first demonstrate how our algorithm adapts an unfair hierarchy into one that achieves fair representation of the protected attributes as desired in the original problem formulation.

In Figures 7.3 and 7.3, we depict the cluster balances of an *unfair* hierarchical clustering algorithm, namely "average-linkage", and subsequently demonstrate that our algorithm effectively concentrates all clusters around the underlying data balance. In particular, we first apply the algorithm and then show how we the balance is further refined by tuning the parameters. The application of Algorithm 13 dramatically improves the representation of the protected attributes in the final clustering and, as such, firmly resolves the problem of achieving fairness.

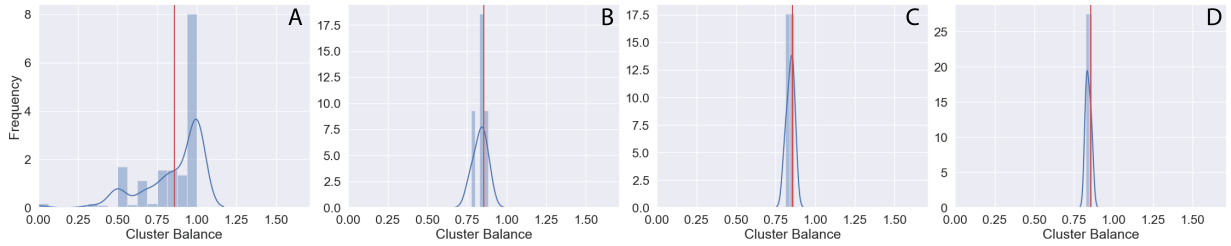


Figure 7.3: Histogram of cluster balances after tree manipulation by Algorithm 13 on a subsample from the *Census* dataset of size $n = 512$. The four panels depict: **(A)** cluster balances after applying the (unfair) average-linkage algorithm, **(B)** the resultant cluster balances after running Algorithm 13 with parameters $(h, k, \varepsilon) = (4, 2, \frac{1}{8 \log_2 n})$, **(C)** cluster balances after tuning $\varepsilon = \frac{1}{4 \log n}$, **(D)** cluster balances after further tuning $\varepsilon = \frac{1}{2 \log n}$. The vertical red line on each plot indicates the balance of the dataset itself.

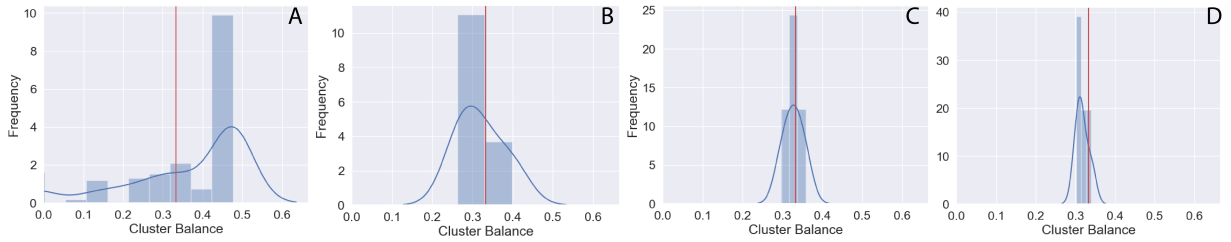
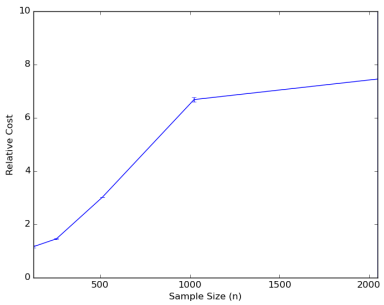


Figure 7.4: Histogram of cluster balances after tree manipulation by Algorithm 13 on a subsample from the *Bank* dataset of size $n = 512$. The four panels depict: **(A)** cluster balances after applying the (unfair) average-linkage algorithm, **(B)** the resultant cluster balances after running Algorithm 13 with parameters $(c, h, k, \varepsilon) = (8, 4, 2, 1/c \cdot \log_2 n)$, **(C)** cluster balances after tuning $c = 4$, **(D)** cluster balances after further tuning $c = 2$. The vertical red line on each plot indicates the balance of the dataset itself.

While reaching this fair partitioning of the data is the overall goal, we further demonstrate that, in modifying the unfair clustering, we only increase the cost approximation by a modest amount. Figure 7.5 illustrates the change in relative cost as we increase the sample size n, the primary influence on our theoretical cost guarantees of Section 7.3. Specifically, we vary n in $\{128, 256, 512, 1024, 2048\}$ and compute 10 replications (on different random seeds) of the fair hierarchical clustering procedure. Figure 7.5 depicts the mean relative cost of these replications with standard error bars. Notably, we see that the cost does increase with n as expected, but the increase relative to the unfair cost obtain by average linkage is only by a small multiplicative



(a) Relative cost comparison as a function of sample size n.

n	[Knittel et al., 2023b]	Our Algorithm
128	3.65240727	1.08586718
256	5.68329859	1.42082465
512	12.30571685	2.5869583
1024	25.17125835	6.6745378
2048	52.93911771	7.86944693

(b) Comparison of relative cost for different values of n.

Figure 7.5: Visualization and numerical comparison of the relative cost of the fair hierarchical clustering algorithm (Algorithm 13) against the unfair baseline. (a) The cost ratio as a function of n. (b) A tabular comparison of the same results.

factor.

As demonstrated through this experimentation, the simplistic procedure of Algorithm 13 not only ensures the desired fairness properties absent in conventional (unfair) clustering algorithms but accomplishes this feat with a negligible rise in the overall cost. These results further highlight the immense value of our work.

7.5 Proofs

This section contains the formal proofs for all of our lemmas and theorems.

7.5.1 Root Splitting and Balancing Proofs

Proof of Lemma 7.3.1. Assume, without loss of generality, that $\delta_1 < \delta_2$ (ie. $\delta = \delta_1$). Let v_{min} and v_{max} be the corresponding nodes for this iteration. By assumption, this implies $\frac{n}{h} - \frac{n_{T'}(v_{min})}{n} \leq$

$\frac{n_{T'}(v_{max})}{n} - \frac{n}{h}$. Furthermore, since the while loop executed, we have that either

$$n_{T'}(v_{max}) = n\left(\frac{1}{h} + \delta_2\right) > n\left(\frac{1}{h} + \epsilon\right) \text{ or } n_{T'}(v_{min}) = n\left(\frac{1}{h} - \delta_1\right) < n\left(\frac{1}{h} - \epsilon\right)$$

Simplification of the above yields $\delta_1 > \epsilon$ or $\delta_2 > \epsilon$. Per our assumption that $\delta_1 < \delta_2$, we proceed to assume that $\delta_2 > \epsilon$ holds.

Observe that $\delta_2 = \frac{n_{T'}(v_{max})}{n} - \frac{1}{h} < \frac{n_{T'}(v_{max})}{n}$. By definition, $\frac{n_{T'}(v_{min})}{n}$ has the largest deviation from $\frac{1}{h}$ of all $v' \in \text{children}(\text{root}(T'))$ with $n_{T'}(v') \leq \frac{n}{h}$. Therefore, for any such v', it holds that $n_{T'}(v') \geq n\left(\frac{1}{h} - \delta_1\right)$. Since $\text{children}(\text{root}(T'))$ forms a total clustering of the data, $\sum_{v' \in \text{children}(\text{root}(T'))} n_{T'}(v') = n$. Combining these facts, we obtain:

$$\begin{aligned}
n &= \sum_{v' \in \text{children}(\text{root}(T'))} n_{T'}(v') \\
&= n_{T'}(v_{max}) + \sum_{v' \in \text{children}(\text{root}(T')) \setminus v_{max}} n_{T'}(v') \\
&\geq n\delta_2 + (h-1) \cdot n\left(\frac{1}{h} - \delta_1\right) + \frac{n}{h} \\
&= n\delta_2 + n - n(h-1)\delta_1
\end{aligned}$$

A final simplification of the above gives $\delta_1 \geq \frac{\delta_2}{h-1}$. Thus, by our assumption on δ_2, we obtain $\delta \geq \frac{\epsilon}{h-1}$. A symmetric analysis for $\delta_2 < \delta_1$ gives a matching result. To see the upper bound, note that the smallest clusters size in the tree is 0, thus $\delta \leq \delta_1 \leq \frac{1}{h}$.

Now, let p denote the parent of v. By the halting condition of the while loop on Line 13, we know $n_{T'}(p) > \delta n$. Since v is the right child of p, it must be the larger of the two children, we have $n_{T'}(v) \geq \frac{n_{T'}(p)}{2} > \frac{\delta n}{2}$, which is at least $\frac{\epsilon}{2(h-1)}$ by the above. Finally, again by the halting

condition on v, we know $n_{T'}(v) \leq \delta n$. $\square$

Proof of Lemma 7.3.2. By definition of the dummy children in execution of the subroutine, the root must have h children.

To analyze the runtime, we examine the decrease in number of leaves for the max child as we step through the procedure. Let $n_{tot} = \sum_{v' \in \text{children} v : n_{T'}(v') > \frac{1}{h}} n_{T'}(v') - \frac{n}{h} \leq n$. The number of vertices in this summation is only reduced, specifically when we move at most δn vertices from the largest to smallest child. The smallest vertex will never exceed $\frac{n}{h}$ by this operation. Further note that v_{max} must be contained in this summation: we necessarily have $n_{T'}(v_{max}) \geq \frac{1}{h}$ with equality holding if and only if *all* children are of size $\frac{1}{h}$ (ie. the algorithm has halted). Moreover, $n_{T'}(v_{max})$ is reduced by at least $\frac{\epsilon n}{2(h-1)}$ in each iteration by Lemma 7.3.1. Therefore, we require at most $2(h-1)$ iterations of the while loop to meet the termination condition. Each iteration involves the traversal down two subtrees for the deletion and insertion operation, each taking $O(n)$ time. The total runtime of the algorithm is thus $O(nh)$.

Finally, assume towards contradiction that the tree is not ϵ-relative balanced with respect to the h children. More specifically, it must be true that there exists some child vertex v' such that either: (1) $n_{T'}(v') \leq \left(\frac{1}{h} - \epsilon\right) n$, or (2) $n_{T'}(v') \geq \left(\frac{1}{h} + \epsilon\right) n$. The first case implies further that $n_{T'}(v_{min}) < \left(\frac{1}{h} - \epsilon\right) n$, violating the termination condition. The second case implies a symmetric contradiction on v_{max}. Thus, the root is ϵ-relatively balanced. $\square$

Proof of Lemma 7.3.3. Let $e = (x, y)$ be an edge separated by the deletion and insertion. This implies that one of the endpoints is in the moved subtree, and the other is not. Without loss of generality, let x be the end point in the moved subtree and let v denote the root of this subtree. By Lemma 7.3.1, $n_T(v) > \frac{\epsilon n}{2(h-1)}$.

Since x is a descendant of v and y is not, their lowest common ancestor, v', must be an ancestor of v. Thus $n_T(v') > n_T(v) > \frac{\epsilon n}{2(h-1)}$, which further implies that

$$\begin{aligned}\text{cost}_T(e) &= w(e) \cdot n_T(v') \\ &\geq w(e) \cdot \frac{n\epsilon}{2(h-1)}\end{aligned}$$

In the modified tree, the maximum cost for this edge is bounded above by $n \cdot w(e)$. Therefore, $\text{cost}_{T'}(e) \leq \frac{2(h-1)}{\epsilon}$, concluding the proof. □

7.5.2 Fair Tree Folding Proofs

Proof of Lemma 7.3.4. For simplicity, assume k^λ divides h. Our algorithm begins by arrange the depth 1 vertices in decreasing order of their fractional representation of the first color, say red. The ordered nodes are then contiguously partitioned into k parts of size $\frac{h}{k}$, and folds vertices of the same index in their respective partitions together. Thus, k clusters are merged.

From the initial h vertices, we have only $\left(\frac{h}{k}\right)^x$ after the $(x-1)$-th fold, where x denotes the current iteration we are at in the folding process.

Let $f(i,j)$ denote the i-th index in the j-th partition of $\mathcal{V}$, i.e., $f(i,j) = jh/k + i$. For every $i \in [h/k]$, we create a new vertex u_i by folding $v_{f(i,j)}$ together for all $j \in k$. Let r_i denote the number of red vertices in u_i. For any i:

$$\begin{aligned}r_i/n_{T'}(u_i) &= \frac{1}{n_{T'}(u_i)} \sum_{j \in [k]} red(v_{f(i,j)}) \\ &\leq \frac{1}{n_{T'}(u_i)} red(v_{f(1,1)}) + \frac{1}{n_{T'}(u_i)} \sum_{j \in \{2,\ldots,k\}} red(v_{f(i,j)})\end{aligned}$$

Note that if we perfectly balanced all cluster sizes at n/h, then $red(v_{f(1,1)}) \leq n/h = n_{T'}(u_i)/k$ would hold. However, $v_{f(1,1)}$ is can be at most a factor of $1+\epsilon$ larger, and $n_{T'}(u_i)$ can be at least a factor of $1-\epsilon$ smaller. Therefore, $\frac{1}{n_{T'}(u_i)} red(v_{f(1,1)}) \leq \frac{1+\epsilon}{k(1-\epsilon)}$.

To bound the second term in our summation, observe that $\frac{red(v_{f(i,j)})}{n_T(v_{f(i,j)})} \leq \frac{red(v_{f(i,j-1)})}{n_T(v_{f(i,j-1)})}$ by definition of our ordering. By the relative balance property, all n_T counts for the vertices are within a factor of $\frac{1+\epsilon}{1-\epsilon}$ of each other. Therefore, $red(v_{f(i,j)}) \leq \frac{1+\epsilon}{1-\epsilon} red(v_{f(i',j-1)})$ for all $i' \in [h/k]$. Since the result holds for all $i' \in [h/k]$, it must also hold for the average of these values:

$$red(v_{f(i,j)}) \leq \frac{k(1+\epsilon)}{h(1-\epsilon)} \sum_{i' \in [h/k]} red(v_{f(i',j-1)}).$$

Returning to the bound on $\frac{r_i}{n_{T'}(u_i)}$, we now have

$$\begin{aligned}
\frac{r_i}{n_{T'}(u_i)} &\leq \frac{1}{n_{T'}(u_i)} red(v_{f(1,1)}) + \frac{1}{n_{T'}(u_i)} \sum_{j \in \{2,\ldots,k\}} red(v_{f(i,j)}) \\
&\leq \frac{1}{n_{T'}(u_i)} red(v_{f(1,1)}) + \frac{k(1+\epsilon)}{h(1-\epsilon) n_{T'}(u_i)} \sum_{j \in \{2,\ldots,k\}} \sum_{i' \in [h/k]} red(v_{f(i',j-1)}) \\
&\leq \frac{1}{n_{T'}(u_i)} red(v_{f(1,1)}) + \frac{k(1+\epsilon)}{h(1-\epsilon) n_{T'}(u_i)} R \\
&\leq \frac{1+\epsilon}{k(1-\epsilon)} + \frac{(1+\epsilon)}{n(1-\epsilon)^2} R \qquad (n_{T'}(u_i) \geq \tfrac{(1-\epsilon)nk}{h}) \\
&= \frac{R}{n} \cdot \frac{1+\epsilon}{(1-\epsilon)^2} \left(1 + \frac{1-\epsilon}{c_R k}\right)
\end{aligned}$$

where $R = O(n)$ is the total number of red points, and c_R is a constant satisfying $R \geq c_R n$.

It remains to lower bound the ratio. We consider the reverse bounds of above. Namely,

$$\begin{aligned}\frac{r_i}{n_{T'}(u_i)} &= \frac{1}{n_{T'}(u_i)} \sum_{j \in [k]} red(v_{f(i,j)}) \\ &\geq \frac{1-\epsilon}{n(1+\epsilon)^2} \sum_{j \in [k-1]} \sum_{i' \in [h/k]} red(v_{f(i',j+1)})\end{aligned}$$

The bound holds by omitting the terms corresponding to $red(v_{f(i',1)})$ in the summation. Using the upper bound of $\frac{(1+\epsilon)kn}{h}$ on the total size of the first partition, we obtain

$$\begin{aligned}\frac{r_i}{n_{T'}(u_i)} &\geq \frac{1-\epsilon}{n(1+\epsilon)^2} \sum_{j \in [k-1]} \sum_{i' \in [h/k]} red(v_{f(i',j+1)}) \\ &= \frac{1-\epsilon}{n(1+\epsilon)^2} \left(R - \sum_{i' \in [h/k]} red(v_{f(i',1)}) \right) \\ &\geq \frac{1-\epsilon}{n(1+\epsilon)^2} \left(R - \frac{(1+\epsilon)kn}{h} \right) \\ &\geq \frac{R}{n} \cdot \frac{1-\epsilon}{(1+\epsilon)^2} \left(1 - \frac{k(1+\epsilon)}{c_R h} \right)\end{aligned}$$

where the final inequality comes from the definition of c_R. Following the processing of this first color, all properties leveraged in the above analysis continue to hold. Concretely, each child of the root still only varies in size by a factor of $(1+\epsilon)/(1-\epsilon)$, and the fairness guarantees of previous colors will always be maintained through any merge. Therefore, the process can be repeated for all colors and the result holds □

Proof of Lemma 7.3.5. We proceed to prove the result by induction. Specifically, we will show

that at the j-th level of recursion, it must hold that

$$\lambda_i \left(\frac{1-\epsilon}{(1+\epsilon)^2} \left(1 - \frac{k(1+\epsilon)}{c_i h} \right) \right)^j \leq \lambda_i^j \leq \lambda_i \left(\frac{1+\epsilon}{(1-\epsilon)^2} \left(1 + \frac{1-\epsilon}{c_i k} \right) \right)^j$$

The result trivially holds for $j = 1$, and we assume it is true at level j.

At level $j+1$, we work within the hierarchy induced on a cluster from the j-th level. By the inductive hypothesis, we have that the number of vertices of color i at level j adheres to

$$\lambda_i \left(\frac{1-\epsilon}{(1+\epsilon)^2} \left(1 - \frac{k(1+\epsilon)}{c_i h} \right) \right)^j \leq \lambda_i^j \leq \lambda_i \left(\frac{1+\epsilon}{(1-\epsilon)^2} \left(1 + \frac{1-\epsilon}{c_i k} \right) \right)^j$$

Using Lemma 7.3.4, we further bound the added multiplicative factor for level $j+1$, yielding the result.

Lastly, we must bound the depth j. At any recursive level, we begin with clusters of size of at most $\frac{(1+\epsilon)n}{h}$ after balancing. We then fold k vertices together at most λ times, for a total size of at most $\frac{(1+\epsilon)nk^\lambda}{h}$. Thus, after the j-th iteration, we have $n \left(\frac{(1+\epsilon)k^\lambda}{h} \right)^j$ vertices left. Upon reaching the level where we have only h vertices left, we must terminate. Setting the above bound equal to h, we conclude

$$j \leq \frac{\log\left(\frac{n}{h}\right)}{\log\left(\frac{h}{(1+\epsilon)k^\lambda}\right)} = O\left(\log\left(\frac{n}{h}\right)\right)$$

provided $h \geq (1+\epsilon)k^\lambda$. Plugging this bound on the depth into our inductive finding finishes the proof. □

Proof of Lemma 7.3.6. An edge separated by the SplitRoot subroutine will incur a cost of at

most $\frac{2(h-1)}{\epsilon}$ by Lemma 7.3.3. Further note that, in Lemma 7.3.3, we consider the worst scenario where $cost_{T'}(e) = n \cdot w(e)$ Therefore, if such an edge is further separated by $\mathrm{MakeFair}$, the cost increase estimate cannot increase.

Let e be an edge first separated by $\mathrm{MakeFair}$. Since the folding only occurs at depth 1 clusters, the cluster containing e must have been at this depth. Therefore, $n_T(e) \geq \frac{(1-\epsilon)n}{h}$ and the max cluster size e belongs to will be at most $\frac{(1+\epsilon)nk^\lambda}{h}$, thus incurring a total cost increase of $\frac{1+\epsilon}{1-\epsilon}k^\lambda$. Combining the two bounds gives the result. □

Proof of Lemma 7.3.7. Let e be an edge separated at some recursive level. Upon separation, in the worst case, the new common ancestor is either the current depth's new root, or one of its children. In the former case, e cannot be involved in further trees down the recursion as its lowest common ancestor is higher in the tree. In the latter case, the edge must be contained in the root of one or more recursive processes. Thus, it cannot be separated any further. □

Proof of Lemma 7.3.8. This follows directly from Lemmas 7.3.6 and 7.3.7. The former bounds the cost of separating an edge at any recursive level, and the latter verifies this happens at most once to each edge. □

Finally, we collect the lemmas to verify our main theorem.

Proof of Theorem 7.3. Applying $\mathrm{SplitRoot}$ guarantees relative balance. Subsequent folding of nodes does not violate this guarantee. The result of Lemma 7.3.8 yields our approximation factor, and Lemma 7.3.5 bounds the proportion of colors at each recursive level. This color bound further yields the fairness of each cluster in the hierarchy, completing the guarantee.

Lemma 7.3.2 demonstrates that $\mathrm{SplitRoot}$ runs in time $O(n'h)$, where n' denotes the current tree size. Execution of $\mathrm{MakeFair}$ is bottle necked by the sorting process required be-

fore folding, running in $O(\lambda n' \log n')$ total time. At any recursive level, a node is involved in at most one recursive instance. Thus, the total time to execute a single recursive level is $O(n(h + \lambda \log n))$. Combining this with the depth bound of Lemma 7.3.5, we have an overall runtime of $O(n \log n(h + \lambda \log n))$. □

7.6 Conclusions, Limitations & Future Works

Fair machine learning strives to combat the limitations of vanilla machine learning by providing a means for bias mitigation for any desired quantifiable bias. However, fair research itself has its own limitations. First, "fairness" can be defined in a number of ways. For instance, [Dwork et al., 2012] explores notions of fairness in classification problems, proposing a type of "individual fairness" which guarantees that similar individuals are treated similarly. This has been extended to clustering by only the work of [Brubach et al., 2020]. Clustering has been predominantly viewed through the lens of "group fairness" which guarantees that different protected classes receive similar, proportional treatment. This was first proposed in clustering by [Chierichetti et al., 2017] and expanded upon in many further works [Ahmadian et al., 2019,**?**,Bercea et al., 2019], including previous fair hierarchical clustering work [Ahmadian et al., 2020a,Knittel et al., 2023b] and this work. Not only is it inherently difficult to account for both of these simultaneously, in some sense these two notions are at odds: if we treat similar individuals similarly, it becomes much harder to impose a diverse range of treatments to individuals in each group, as they often are quite similar themselves. This illustrates the necessity of applying fair algorithms on a case by case basis, carefully considering what fair effect is most desirable.

Second, bias mitigation through fair algorithmic techniques has been shown to cause harm

in at least one application [Ben-Porat et al., 2021]. Thus, all fair machine learning techniques, including ours, should be used with great caution and consideration of all downstream effects. We defer the reader to [Barocas et al., 2019] as well as the Fair Clustering Tutorial [AAAI 2023] for further perspectives on fair machine learning and its limitations.

The main results of this paper are theoretical guarantees on algorithmic performance. Naturally, this provides additional limitations, predominantly in that the guarantees only hold under the assumptions clearly stated in this paper. For instance, our main algorithm requires that each color represents a constant fraction of the total data. This assumption is quite realistic and can be found throughout fair learning literature, but there are certain practical instances where our results may not be applicable. In addition, since our proofs only consider worst-case analysis, we do not know much about the average-case guarantees of our algorithms (other than they are strictly better than the worst case). We account for this through empirical evaluation, though this is inherently limited as tested data sets cannot represent all potential applications.

Finally, our work focuses on the cost objective function. While cost is highly regarded by the hierarchical clustering community [Dasgupta, 2016], it may not be an appropriate metric for all applications. Moreover, it is sometimes viewed as impractical in that it is quite difficult to provide worst-case guarantees for [Charikar and Chatziafratis, 2017]. Future work might consider evaluating our algorithms using other objectives such as revenue [Moseley and Wang, 2017] or value [Cohen-Addad et al., 2018] to see how they perform.

Chapter 8: Open Directions & Future Work

In this thesis, I have demonstrated that, across a diverse array of algorithmic domains, it is possible to design solutions that equitably treat users while incurring only a minimal cost to efficiency. Contrary to the notion that fairness necessarily demands significant trade-offs, my results indicate that fairness constraints can be incorporated into algorithmic design in a principled way, often with only a small impact on computational complexity or solution quality.

Offline Resource Allocation. For the domain of fair division, I advanced the state of research on EFX, a long-standing open problem in algorithmic game theory. By introducing two novel formulations of EFX, I provided new structural insights that offer a fresh perspective on a problem that has remained largely stagnant. These contributions not only extend the theoretical understanding of fair allocation but also lay the groundwork for future algorithmic advancements that may bring researchers closer to resolving the standard EFX problem. The results presented here suggest that fairness guarantees in discrete allocation settings can be meaningfully improved, even when traditional solution concepts appear intractable.

In this problem domain, the most pressing open question posed by my results is:

Problem 8.0.1. *Does an $O(w)$-approximation algorithm exist for the WEFX problem for any number of agents $n \geq 2$? Can this approximation factor be further improved to a constant $O(1)$?*

Specifically, any result which interpolates between the weighted and unweighted problem

instances would be an interesting leap forward.

Online Resource Allocation. In the realm of online algorithm design, I introduced fairness constraints into well-established problems, expanding the applicability of fair division principles beyond static settings. Specifically, I initiated the study of the Santa Claus problem under online conditions, addressing fairness considerations in resource allocation when decisions must be made sequentially with incomplete information.

Problem 8.0.2. *What is the optimal approximation ratio for the online Santa Claus problem? In particular, can it be tightly characterized as* $\Theta\left(\frac{\log n}{\epsilon^2}\right)$*, or can it be further improved to* $\Theta\left(\frac{\log n}{\epsilon}\right)$*?*

Additionally, I developed the first randomized algorithms for fair online bipartite matching, demonstrating that fairness-aware allocations can be achieved even under stringent real-time constraints. These results contribute to a growing understanding of how fairness can be integrated into dynamic decision-making, opening new directions for future research in competitive online environments.

Problem 8.0.3. *What is the true price of fairness for the online matching problem? Moreover, can the well-known optimal "Ranking" algorithm be successfully adapted or generalized to address fairness constraints?*

In recent discussions with collaborators, I have demonstrated that the price of fairness may absolutely interpolate between

Hierarchical Clustering. For the problem of hierarchical clustering, I significantly reduced the gap between fair and optimal solutions by designing a flexible procedure that can be applied to any unfair clustering hierarchy. This method, which preserves interpretability while ensuring demographic representation, improves upon previous fairness-aware clustering approaches,

demonstrating that fairness need not come at a prohibitive computational cost. While my results establish a polylogarithmic overhead for ensuring fairness in hierarchical clustering, an open question remains:

Problem 8.0.4. *Given an unfair clustering of n data points, what is the smallest possible increase in clustering cost required to achieve fairness constraints?*

I conjecture that a logarithmic increase must persist and that our result is tight. However, an $O(1)$ minimal cost would suggest that there is no real cost to being fair—a considerably more inspiring result.

Future Work. The broader goal of this thesis has been to demonstrate that fairness constraints can be incorporated into optimization and machine learning algorithms without rendering them computationally impractical. The results presented here challenge the assumption that fairness inherently conflicts with efficiency, instead showing that structured algorithmic interventions can mitigate disparities without drastically increasing resource demands.

However, while these theoretical contributions offer promising directions, the real-world resolution of unfair outcomes arising from algorithmic decision-making remains an ongoing challenge. Algorithmic fairness is a deeply interdisciplinary problem, intersecting with economics, law, and ethics, and fully addressing it requires frameworks that go beyond purely mathematical formulations. Future work should explore how the theoretical guarantees established here translate to practical deployment, particularly in high-stakes settings such as medical resource allocation, hiring algorithms, and algorithmic governance.

Ultimately, I hope that this work serves as a stepping stone toward the broader goal of designing algorithms that not only optimize for efficiency but also respect fundamental fair-

ness principles. By demonstrating that fair and efficient solutions can coexist, I aim to inspire researchers and practitioners to further investigate these challenges, ensuring that algorithmic decision-making is both powerful and just.

Bibliography

[Agrawal and Devanur, 2014] Agrawal, S. and Devanur, N. R. (2014). Fast algorithms for online stochastic convex programming. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1405–1424. SIAM.

[Ahmadian et al., 2020a] Ahmadian, S., Epasto, A., Knittel, M., Kumar, R., Mahdian, M., Moseley, B., Pham, P., Vassilvitskii, S., and Wang, Y. (2020a). Fair hierarchical clustering. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[Ahmadian et al., 2019] Ahmadian, S., Epasto, A., Kumar, R., and Mahdian, M. (2019). Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 267–275.

[Ahmadian et al., 2020b] Ahmadian, S., Epasto, A., Kumar, R., and Mahdian, M. (2020b). Fair correlation clustering. In *International conference on artificial intelligence and statistics*, pages 4195–4205. PMLR.

[Akrami et al., 2023] Akrami, H., Alon, N., Chaudhury, B. R., Garg, J., Mehlhorn, K., and Mehta, R. (2023). EFX: A simpler approach and an (almost) optimal guarantee via rainbow cycle number. In *Proceedings of the 24th ACM Conference on Economics and Computation*, EC '23, page 61, New York, NY, USA. Association for Computing Machinery.

[Akrami and Garg, 2024] Akrami, H. and Garg, J. (2024). Breaking the 3/4 barrier for approximate maximin share. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA, 2024*.

[Akrami et al., 2022] Akrami, H., Rezvan, R., and Seddighin, M. (2022). An EF2X allocation protocol for restricted additive valuations. In Raedt, L. D., editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 17–23. ijcai.org.

[Albers, 1997] Albers, S. (1997). Better bounds for online scheduling. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 130–139.

[Aleksandrov et al., 2015] Aleksandrov, M., Aziz, H., Gaspers, S., and Walsh, T. (2015). Online fair division: analysing a food bank problem. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 2540–2546.

[Aleksandrov and Walsh, 2020] Aleksandrov, M. and Walsh, T. (2020). Online fair division: A survey. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13557–13562.

[Amanatidis et al., 2023] Amanatidis, G., Aziz, H., Birmpas, G., Filos-Ratsikas, A., Li, B., Moulin, H., Voudouris, A. A., and Wu, X. (2023). Fair division of indivisible goods: Recent progress and open questions. *Artificial Intelligence*, page 103965.

[Amanatidis et al., 2021] Amanatidis, G., Birmpas, G., Filos-Ratsikas, A., Hollender, A., and Voudouris, A. A. (2021). Maximum Nash welfare and other stories about EFX. *Theoretical Computer Science*, 863:69–85.

[Amanatidis et al., 2022] Amanatidis, G., Birmpas, G., Filos-Ratsikas, A., and Voudouris, A. A. (2022). Fair division of indivisible goods: A survey. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022*, pages 5385–5393.

[Amanatidis et al., 2017] Amanatidis, G., Markakis, E., Nikzad, A., and Saberi, A. (2017). Approximation algorithms for computing maximin share allocations. *ACM Transactions on Algorithms (TALG)*, 13(4):1–28.

[Amanatidis et al., 2020] Amanatidis, G., Markakis, E., and Ntokos, A. (2020). Multiple birds with one stone: Beating 1/2 for EFX and GMMS via envy cycle elimination. *Theoretical Computer Science*, 841:94–109.

[Anari et al., 2016] Anari, N., Gharan, S. O., Saberi, A., and Singh, M. (2016). Nash social welfare, matrix permanent, and stable polynomials. *arXiv preprint arXiv:1609.07056*.

[Anari et al., 2018] Anari, N., Mai, T., Gharan, S. O., and Vazirani, V. V. (2018). Nash social welfare for indivisible items under separable, piecewise-linear concave utilities. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2274–2290. SIAM.

[Angwin et al., 2016a] Angwin, J., Larson, J., and Kirchner, L. (2016a). Machine bias. *ProPublica*.

[Angwin et al., 2016b] Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016b). Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks.

[Annamalai et al., 2017] Annamalai, C., Kalaitzis, C., and Svensson, O. (2017). Combinatorial algorithm for restricted max-min fair allocation. *ACM Transactions on Algorithms (TALG)*, 13(3):1–28.

[Apostol, 1999] Apostol, T. M. (1999). An elementary view of euler's summation formula. *The American Mathematical Monthly*, 106(5):409–418.

[Arifin and Asano, 2006] Arifin, A. Z. and Asano, A. (2006). Image segmentation by histogram thresholding using hierarchical cluster analysis. *Pattern Recognit. Lett.*, 27(13):1515–1521.

[Asadpour et al., 2012] Asadpour, A., Feige, U., and Saberi, A. (2012). Santa claus meets hypergraph matchings. *ACM Trans. Algorithms*, 8(3).

[Asadpour and Saberi, 2010] Asadpour, A. and Saberi, A. (2010). An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM Journal on Computing*, 39(7):2970–2989.

[Azar and Richter, 2004] Azar, Y. and Richter, Y. (2004). The zero-one principle for switching networks. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 64–71.

[Aziz et al., 2022] Aziz, H., Caragiannis, I., Igarashi, A., and Walsh, T. (2022). Fair allocation of indivisible goods and chores. *Autonomous Agents and Multi-Agent Systems*, 36:1–21.

[Aziz et al., 2019] Aziz, H., Chan, H., and Li, B. (2019). Weighted maxmin fair share allocation of indivisible chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 46–52. AAAI Press.

[Aziz et al., 2020] Aziz, H., Moulin, H., and Sandomirskiy, F. (2020). A polynomial-time algorithm for computing a pareto optimal and almost proportional allocation. *Operations Research Letters*, 48(5):573–578.

[Aziz et al., 2017] Aziz, H., Rauchecker, G., Schryen, G., and Walsh, T. (2017). Algorithms for max-min share fair allocation of indivisible chores. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

[Babaioff et al., 2021] Babaioff, M., Ezra, T., and Feige, U. (2021). Fair and truthful mechanisms for dichotomous valuations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5119–5126.

[Babaioff et al., 2024] Babaioff, M., Ezra, T., and Feige, U. (2024). Fair-share allocations for agents with arbitrary entitlements. *Mathematics of Operations Research*, 49(4):2180–2211.

[Babaioff et al., 2007] Babaioff, M., Immorlica, N., Kempe, D., and Kleinberg, R. (2007). A knapsack secretary problem with applications. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, pages 16–28. Springer.

[Backurs et al., 2019] Backurs, A., Indyk, P., Onak, K., Schieber, B., Vakilian, A., and Wagner, T. (2019). Scalable fair clustering. In *International conference on machine learning*, pages 405–413. PMLR.

[Balseiro and Gur, 2019] Balseiro, S. R. and Gur, Y. (2019). Learning in repeated auctions with budgets: Regret minimization and equilibrium. *Management Science*, 65(9):3952–3968.

[Bamas et al., 2025] Bamas, É., Morell, S., and Rohwedder, L. (2025). The submodular santa claus problem. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 616–640. SIAM.

[Banerjee and Johari, 2019] Banerjee, S. and Johari, R. (2019). Ride sharing. *Sharing Economy: Making Supply Meet Demand*, pages 73–97.

[Bansal and Sviridenko, 2006] Bansal, N. and Sviridenko, M. (2006). The santa claus problem. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 31–40, New York, NY, USA. Association for Computing Machinery.

[Barbanel, 1996] Barbanel, J. (1996). Game-theoretic algorithms for fair and strongly fair cake division with entitlements. In *Colloquium Mathematicae*, volume 69, pages 59–73.

[Barman and Krishnamurthy, 2019] Barman, S. and Krishnamurthy, S. K. (2019). On the proximity of markets with integral equilibria. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1748–1755.

[Barman and Krishnamurthy, 2020] Barman, S. and Krishnamurthy, S. K. (2020). Approximation algorithms for maximin fair division. *ACM Transactions on Economics and Computation (TEAC)*, 8(1):1–28.

[Barman et al., 2018] Barman, S., Krishnamurthy, S. K., and Vaish, R. (2018). Finding fair and efficient allocations. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC)*, pages 557–574.

[Barocas et al., 2019] Barocas, S., Hardt, M., and Narayanan, A. (2019). *Fairness and Machine Learning*. `www.fairmlbook.org`.

[Bei et al., 2021] Bei, X., Lu, X., Manurangsi, P., and Suksompong, W. (2021). The price of fairness for indivisible goods. *Theory of Computing Systems*, 65:1069–1093.

[Ben-Porat et al., 2021] Ben-Porat, O., Sandomirskiy, F., and Tennenholtz, M. (2021). Protecting the protected group: Circumventing harmful fairness. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 5176–5184. AAAI Press.

[Benade et al., 2018] Benade, G., Kazachkov, A. M., Procaccia, A. D., and Psomas, C.-A. (2018). How to make envy vanish over time. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 593–610.

[Bera et al., 2019] Bera, S., Chakrabarty, D., Flores, N., and Negahbani, M. (2019). Fair algorithms for clustering. *Advances in Neural Information Processing Systems*, 32.

[Bercea et al., 2019] Bercea, I. O., Groß, M., Khuller, S., Kumar, A., Rösner, C., Schmidt, D. R., and Schmidt, M. (2019). On the cost of essentially fair clusterings. In *APPROX-RANDOM*, pages 18:1–18:22.

[Berendsohn et al., 2022] Berendsohn, B. A., Boyadzhiyska, S., and Kozma, L. (2022). Fixed-point cycles and approximate EFX allocations. In *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*, page 17. Schloss Dagstuhl.

[Berger et al., 2022] Berger, B., Cohen, A., Feldman, M., and Fiat, A. (2022). Almost full EFX exists for four agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4826–4833.

[Bertsimas et al., 2011] Bertsimas, D., Farias, V. F., and Trichakis, N. (2011). The price of fairness. *Operations research*, 59(1):17–31.

[Bezáková and Dani, 2005] Bezáková, I. and Dani, V. (2005). Allocating indivisible goods. *ACM SIGecom Exchanges*, 5(3):11–18.

[Blanchard et al., 2021] Blanchard, P., Higham, D. J., and Higham, N. J. (2021). Accurately computing the log-sum-exp and softmax functions. *IMA Journal of Numerical Analysis*, 41(4):2311–2330.

[Blum et al., 2006] Blum, A., Sandholm, T., and Zinkevich, M. (2006). Online algorithms for market clearing. *Journal of the ACM (JACM)*, 53(5):845–879.

[Bogen and Rieke, 2018] Bogen, M. and Rieke, A. (2018). Help wanted: An examination of hiring algorithms, equity, and bias. Technical report, Upturn.

[Borodin and El-Yaniv, 2005] Borodin, A. and El-Yaniv, R. (2005). *Online computation and competitive analysis*. cambridge university press.

[Bouveret et al., 2016] Bouveret, S., Chevaleyre, Y., and Maudet, N. (2016). Fair allocation of indivisible goods. In *Handbook of Computational Social Choice*.

[Bouveret and Lemaître, 2016] Bouveret, S. and Lemaître, M. (2016). Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems*, 30(2):259–290.

[Brams and Taylor, 1995] Brams, S. J. and Taylor, A. D. (1995). An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18.

[Brams and Taylor, 1996] Brams, S. J. and Taylor, A. D. (1996). *Fair division - from cake-cutting to dispute resolution*. Cambridge University Press.

[Brubach et al., 2020] Brubach, B., Chakrabarti, D., Dickerson, J. P., Khuller, S., Srinivasan, A., and Tsepenekas, L. (2020). A pairwise fair and community-preserving approach to k-center clustering. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1178–1189. PMLR.

[Buchbinder et al., 2007] Buchbinder, N., Jain, K., and Naor, J. S. (2007). Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264. Springer.

[Budish, 2011] Budish, E. (2011). The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103.

[Budish and Cantillon, 2010] Budish, E. and Cantillon, E. (2010). The multi-unit assignment problem: Theory and evidence from course allocation at Harvard. *American Economic Review*, 102.

[Burrell and Fourcade, 2021] Burrell, J. and Fourcade, M. (2021). The society of algorithms. *Annual Review of Sociology*, 47(1):213–237.

[Cai et al., 2004] Cai, D., He, X., Li, Z., Ma, W., and Wen, J. (2004). Hierarchical clustering of WWW image search results using visual, textual and link information. In *Proceedings of the 12th ACM International Conference on Multimedia*, pages 952–959. ACM.

[Caragiannis et al., 2019a] Caragiannis, I., Gravin, N., and Huang, X. (2019a). Envy-freeness up to any item with high Nash welfare: The virtue of donating items. In *Proceedings of the 20th ACM Conference on Economics and Computation (EC)*, pages 527–545.

[Caragiannis et al., 2012] Caragiannis, I., Kaklamanis, C., Kanellopoulos, P., and Kyropoulou, M. (2012). The efficiency of fair division. *Theory of Computing Systems*, 50:589–610.

[Caragiannis et al., 2019b] Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A. D., Shah, N., and Wang, J. (2019b). The unreasonable fairness of maximum Nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–32.

[Chakrabarti et al., 2022] Chakrabarti, D., Dickerson, J. P., Esmaeili, S. A., Srinivasan, A., and Tsepenekas, L. (2022). A new notion of individually fair clustering: α-equitable k-center. In *International conference on artificial intelligence and statistics*, pages 6387–6408. PMLR.

[Chakrabarty et al., 2009] Chakrabarty, D., Chuzhoy, J., and Khanna, S. (2009). On allocating goods to maximize fairness. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 107–116. IEEE Computer Society.

[Chakraborty et al., 2020] Chakraborty, M., Suksompong, W., and Zick, Y. (2020). Weighted envy-freeness in indivisible item allocation. *ACM Transactions on Economics and Computation (TEAC)*, 9:1 – 39.

[Chan et al., 2019] Chan, H., Chen, J., Li, B., and Wu, X. (2019). Maximin-aware allocations of indivisible goods. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 137–143. AAAI Press.

[Charikar and Chatziafratis, 2017] Charikar, M. and Chatziafratis, V. (2017). Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 841–854. SIAM.

[Charikar et al., 2019] Charikar, M., Chatziafratis, V., Niazadeh, R., and Yaroslavtsev, G. (2019). Hierarchical clustering for euclidean data. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2721–2730. PMLR.

[Chatziafratis et al., 2018] Chatziafratis, V., Niazadeh, R., and Charikar, M. (2018). Hierarchical clustering with structural constraints. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 773–782. PMLR.

[Chaudhury et al., 2020] Chaudhury, B. R., Garg, J., and Mehlhorn, K. (2020). EFX exists for three agents. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 1–19.

[Chaudhury et al., 2021a] Chaudhury, B. R., Garg, J., Mehlhorn, K., Mehta, R., and Misra, P. (2021a). Improving EFX guarantees through rainbow cycle number. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC)*, pages 310–311. ACM.

[Chaudhury et al., 2021b] Chaudhury, B. R., Kavitha, T., Mehlhorn, K., and Sgouritsa, A. (2021b). A little charity guarantees almost envy-freeness. *SIAM J. Comput.*, 50(4):1336–1358.

[Chayka, 2025] Chayka, K. (2025). *Filterworld: How algorithms flattened culture*. Vintage Books.

[Chen et al., 2020a] Chen, J., Zhang, S., He, X., Lin, Q., Zhang, H., Hao, D., Kang, Y., Gao, F., Xu, Z., Dang, Y., et al. (2020a). How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 373–384.

[Chen et al., 2019] Chen, X., Fain, B., Lyu, L., and Munagala, K. (2019). Proportionally fair clustering. In *International conference on machine learning*, pages 1032–1041. PMLR.

[Chen and Qin, 2011] Chen, X. and Qin, S. (2011). On-line machine covering on two machines with local migration. *Computers & Mathematics with Applications*, 62(5):2336–2341.

[Chen et al., 2020b] Chen, Y., Yang, X., Dong, H., He, X., Zhang, H., Lin, Q., Chen, J., Zhao, P., Kang, Y., Gao, F., et al. (2020b). Identifying linked incidents in large-scale online service systems. In *Proceedings of the 28th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, pages 304–314.

[Cheng and Mao, 2018] Cheng, S.-W. and Mao, Y. (2018). Restricted Max-Min Fair Allocation. In Chatzigiannakis, I., Kaklamanis, C., Marx, D., and Sannella, D., editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:13, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Chhabra and Mohapatra, 2022] Chhabra, A. and Mohapatra, P. (2022). Fair algorithms for hierarchical agglomerative clustering. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 206–211. IEEE.

[Chierichetti et al., 2017] Chierichetti, F., Kumar, R., Lattanzi, S., and Vassilvitskii, S. (2017). Fair clustering through fairlets. In *NIPS*, pages 5029–5037.

[Christodoulou et al., 2023] Christodoulou, G., Fiat, A., Koutsoupias, E., and Sgouritsa, A. (2023). Fair allocation in graphs. In *Proceedings of the 24th ACM Conference on Economics and Computation*, EC '23, pages 473–488, New York, NY, USA. Association for Computing Machinery.

[Cohen-Addad et al., 2017] Cohen-Addad, V., Kanade, V., and Mallmann-Trenn, F. (2017). Hierarchical clustering beyond the worst-case. In *NIPS*, pages 6201–6209.

[Cohen-Addad et al., 2018] Cohen-Addad, V., Kanade, V., Mallmann-Trenn, F., and Mathieu, C. (2018). Hierarchical clustering: Objective functions and algorithms. In *SODA*, pages 378–397.

[Cole et al., 2017] Cole, R., Devanur, N., Gkatzelis, V., Jain, K., Mai, T., Vazirani, V. V., and Yazdanbod, S. (2017). Convex program duality, fisher markets, and Nash social welfare. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 459–460.

[Cole and Gkatzelis, 2015] Cole, R. and Gkatzelis, V. (2015). Approximating the Nash social welfare with indivisible items. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 371–380.

[Conitzer et al., 2017] Conitzer, V., Freeman, R., and Shah, N. (2017). Fair public decision making. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 629–646.

[Conitzer et al., 2021] Conitzer, V., Kroer, C., Sodomka, E., and Stier-Moses, N. E. (2021). Multiplicative pacing equilibria in auction markets. *Operations Research*.

[Cseh and Fleiner, 2020] Cseh, Á. and Fleiner, T. (2020). The complexity of cake cutting with unequal shares. *ACM Transactions on Algorithms (TALG)*, 16(3):1–21.

[Dasgupta, 2016] Dasgupta, S. (2016). A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 118–127.

[Davies et al., 2020] Davies, S., Rothvoss, T., and Zhang, Y. (2020). A tale of santa claus, hypergraphs and matroids. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2748–2757. SIAM.

[Dehghani et al., 2018] Dehghani, S., Farhadi, A., HajiAghayi, M., and Yami, H. (2018). Envy-free chore division for an arbitrary number of agents. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2564–2583. SIAM.

[Devanur and Hayes, 2009] Devanur, N. R. and Hayes, T. P. (2009). The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 71–78.

[Dickerson et al., 2021] Dickerson, J. P., Sankararaman, K. A., Srinivasan, A., and Xu, P. (2021). Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. *ACM Transactions on Economics and Computation (TEAC)*, 9(3):1–17.

[Dua and Graff, 2017] Dua, D. and Graff, C. (2017). UCI machine learning repository.

[Dwork et al., 2012] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. S. (2012). Fairness through awareness. In Goldwasser, S., editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 214–226. ACM.

[Eisenberg and Gale, 1959] Eisenberg, E. and Gale, D. (1959). Consensus of subjective probabilities: The pari-mutuel method. *The Annals of Mathematical Statistics*, 30(1):165–168.

[Epstein et al., 2011] Epstein, L., Levin, A., and van Stee, R. (2011). Max-min online allocations with a reordering buffer. *SIAM Journal on Discrete Mathematics*, 25(3):1230–1250.

[Esmaeili et al., 2021] Esmaeili, S. A., Brubach, B., Srinivasan, A., and Dickerson, J. (2021). Fair clustering under a bounded cost. In *Advances in Neural Information Processing Systems*, pages 14345–14357.

[Esmaeili et al., 2020] Esmaeili, S. A., Brubach, B., Tsepenekas, L., and Dickerson, J. (2020). Probabilistic fair clustering. In *Advances in Neural Information Processing Systems*.

[Etkin et al., 2005] Etkin, R., Parekh, A., and Tse, D. (2005). Spectrum sharing for unlicensed bands. In *In Proceedings of the first IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks*.

[Fallucchi et al., 2021] Fallucchi, F., Faravelli, M., and Quercia, S. (2021). Fair allocation of scarce medical resources in the time of covid-19: what do people think? *Journal of medical ethics*, 47(1):3–6.

[Farhadi et al., 2019] Farhadi, A., Ghodsi, M., Hajiaghayi, M. T., Lahaie, S., Pennock, D., Seddighin, M., Seddighin, S., and Yami, H. (2019). Fair allocation of indivisible goods to asymmetric agents. *Journal of Artificial Intelligence Research*, 64:1–20.

[Farhadi et al., 2021] Farhadi, A., Hajiaghayi, M., Latifian, M., Seddighin, M., and Yami, H. (2021). Almost envy-freeness, envy-rank, and Nash social welfare matchings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5355–5362.

[Feige, 2008] Feige, U. (2008). On allocations that maximize fairness. In *SODA*, volume 8, pages 287–293. Citeseer.

[Feige et al., 2021] Feige, U., Sapir, A., and Tauber, L. (2021). A tight negative example for MMS fair allocations. In *International Conference on Web and Internet Economics*, pages 355–372. Springer.

[Feldman et al., 2010] Feldman, J., Henzinger, M., Korula, N., Mirrokni, V. S., and Stein, C. (2010). Online stochastic packing applied to display ad allocation. In *European Symposium on Algorithms*, pages 182–194. Springer.

[Feldman et al., 2009a] Feldman, J., Korula, N., Mirrokni, V., Muthukrishnan, S., and Pál, M. (2009a). Online ad assignment with free disposal. In *International workshop on internet and network economics*, pages 374–385. Springer.

[Feldman et al., 2009b] Feldman, J., Mehta, A., Mirrokni, V., and Muthukrishnan, S. (2009b). Online stochastic matching: Beating 1-1/e. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126. IEEE.

[Feldman et al., 2024] Feldman, M., Mauras, S., and Ponitka, T. (2024). On optimal tradeoffs between EFX and Nash welfare. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9688–9695.

[Ferragina and Gulli, 2005] Ferragina, P. and Gulli, A. (2005). A personalized search engine based on web-snippet hierarchical clustering. In *Proceedings of the 14th international conference on World Wide Web*, pages 801–810. ACM.

[Foley, 1967] Foley, D. (1967). Resource allocation and the public sector. *Yale Econ Essays*, 7(1):45–98.

[Gao and Pavel, 2017] Gao, B. and Pavel, L. (2017). On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*.

[Garg et al., 2019] Garg, J., McGlaughlin, P., and Taki, S. (2019). Approximating maximin share allocations. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*, pages 20–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[Garg and Murhekar, 2023] Garg, J. and Murhekar, A. (2023). Computing fair and efficient allocations with few utility values. *Theoretical Computer Science*, 962:113932.

[Garg and Taki, 2020] Garg, J. and Taki, S. (2020). An improved approximation algorithm for maximin shares. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 379–380.

[Ghodsi et al., 2018] Ghodsi, M., HajiAghayi, M., Seddighin, M., Seddighin, S., and Yami, H. (2018). Fair allocation of indivisible goods: Improvements and generalizations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 539–556.

[Gillespie, 2016] Gillespie, T. (2016). # trendingistrending: When algorithms become culture. In *Algorithmic cultures*, pages 64–87. Routledge.

[Goel and Mehta, 2008] Goel, G. and Mehta, A. (2008). Online budgeted matching in random input models with applications to adwords. In *SODA*, volume 8, pages 982–991.

[Goldman and Procaccia, 2015] Goldman, J. and Procaccia, A. D. (2015). Spliddit: Unleashing fair division algorithms. *ACM SIGecom Exchanges*, 13(2):41–46.

[Gollapudi and Panigrahi, 2014] Gollapudi, S. and Panigrahi, D. (2014). Fair allocation in online markets. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1179–1188.

[Gorokh et al., 2020] Gorokh, A., Banerjee, S., Jin, B., and Gkatzelis, V. (2020). Online Nash social welfare via promised utilities. *arXiv e-prints*, pages arXiv–2008.

[Gourvès and Monnot, 2019] Gourvès, L. and Monnot, J. (2019). On maximin share allocations in matroids. *Theoretical Computer Science*, 754:50–64.

[Guo et al., 2023] Guo, H., Li, W., and Deng, B. (2023). A survey on fair allocation of chores. *Mathematics*, 11(16):3616.

[Gupta and Molinaro, 2014] Gupta, A. and Molinaro, M. (2014). How experts can solve lps online. In *European Symposium on Algorithms*, pages 517–529. Springer.

[Hajiaghayi et al., 2024] Hajiaghayi, M., Jahan, S., Sharifi, M., Shin, S., and Springer, M. (2024). Fairness and efficiency in online class matching. *Advances in Neural Information Processing Systems*, 37:18416–18444.

[Hajiaghayi et al., 2007] Hajiaghayi, M. T., Kleinberg, R., and Sandholm, T. (2007). Automated online mechanism design and prophet inequalities. In *AAAI*, volume 7, pages 58–65.

[He and Jiang, 2005] He, Y. and Jiang, Y. (2005). Optimal semi-online preemptive algorithms for machine covering on two uniform machines. *Theoretical Computer Science*, 339(2-3):293–314.

[Hosseini et al., 2024] Hosseini, H., Huang, Z., Igarashi, A., and Shah, N. (2024). Class fairness in online matching. *Artificial Intelligence*, 335:104177.

[Hosseini and Searns, 2021] Hosseini, H. and Searns, A. (2021). Guaranteeing maximin shares: Some agents left behind. In Zhou, Z.-H., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 238–244. International Joint Conferences on Artificial Intelligence Organization. Main Track.

[Hosseini et al., 2022] Hosseini, H., Searns, A., and Segal-Halevi, E. (2022). Ordinal maximin share approximation for goods. *Journal of Artificial Intelligence Research*, 74:353–391.

[Huang and Shu, 2021] Huang, Z. and Shu, X. (2021). Online stochastic matching, poisson arrivals, and the natural linear program. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 682–693.

[Huang et al., 2019] Huang, Z., Tang, Z. G., Wu, X., and Zhang, Y. (2019). Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals. *ACM Transactions on Algorithms (TALG)*, 15(3):1–15.

[Igarashi and Yokoyama, 2023] Igarashi, A. and Yokoyama, T. (2023). Kajibuntan: a house chore division app. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 16449–16451.

[Jahan et al., 2023] Jahan, S. C., Seddighin, M., Seyed-Javadi, S.-M., and Sharifi, M. (2023). Rainbow cycle number and EFX allocations: (almost) closing the gap. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23.

[Jansen and Rohwedder, 2020] Jansen, K. and Rohwedder, L. (2020). A note on the integrality gap of the configuration lp for restricted santa claus. *Information Processing Letters*, 164:106025.

[Jin and Williamson, 2021] Jin, B. and Williamson, D. P. (2021). Improved analysis of ranking for online vertex-weighted bipartite matching in the random order model. In *International Conference on Web and Internet Economics*, pages 207–225. Springer.

[Kalyanasundaram and Pruhs, 2000] Kalyanasundaram, B. and Pruhs, K. R. (2000). An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1-2):319–325.

[Karande et al., 2011] Karande, C., Mehta, A., and Tripathi, P. (2011). Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 587–596.

[Karp et al., 1990] Karp, R. M., Vazirani, U. V., and Vazirani, V. V. (1990). An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358.

[Kelly et al., 1998] Kelly, F. P., Maulloo, A. K., and Tan, D. K. H. (1998). Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252.

[Kesselheim et al., 2014] Kesselheim, T., Tönnis, A., Radke, K., and Vöcking, B. (2014). Primal beats dual on online packing lps in the random-order model. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 303–312.

[Kleinberg, 2005] Kleinberg, R. (2005). A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 630–631. Citeseer.

[Kleindessner et al., 2019a] Kleindessner, M., Awasthi, P., and Morgenstern, J. (2019a). Fair k-center clustering for data summarization. In *ICML*, pages 3448–3457.

[Kleindessner et al., 2019b] Kleindessner, M., Samadi, S., Awasthi, P., and Morgenstern, J. (2019b). Guarantees for spectral clustering with fairness constraints. In *ICML*, pages 3458–3467.

[Knittel et al., 2023a] Knittel, M., Springer, M., Dickerson, J., and Hajiaghayi, M. (2023a). Fair, polylog-approximate low-cost hierarchical clustering. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 44836–44846. Curran Associates, Inc.

[Knittel et al., 2023b] Knittel, M., Springer, M., Dickerson, J. P., and Hajiaghayi, M. (2023b). Generalized reductions: Making any hierarchical clustering fair and balanced with low cost.

[Kou and Lou, 2012] Kou, G. and Lou, C. (2012). Multiple factor hierarchical clustering algorithm for large scale web page and search engine clickstream data. *Ann. Oper. Res.*, 197(1):123–134.

[Kraskov et al., 2003] Kraskov, A., Stögbauer, H., Andrzejak, R. G., and Grassberger, P. (2003). Hierarchical clustering using mutual information. *CoRR*, q-bio.QM/0311037.

[Kurokawa et al., 2018] Kurokawa, D., Procaccia, A. D., and Wang, J. (2018). Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM (JACM)*, 65(2):1–27.

[Kurtz, 1970] Kurtz, T. G. (1970). Solutions of ordinary differential equations as limits of pure jump markov processes. *Journal of applied Probability*, 7(1):49–58.

[Latzer et al., 2016] Latzer, M., Hollnbuchner, K., Just, N., and Saurwein, F. (2016). The economics of algorithmic selection on the internet. In *Handbook on the Economics of the Internet*, pages 395–425. Edward Elgar Publishing.

[Laventhal et al., 2020] Laventhal, N., Basak, R., Dell, M. L., Diekema, D., Elster, N., Geis, G., Mercurio, M., Opel, D., Shalowitz, D., Statter, M., et al. (2020). The ethics of creating a resource allocation strategy during the covid-19 pandemic. *Pediatrics*, 146(1).

[Ledford, 2019] Ledford, H. (2019). Millions of black people affected by racial bias in healthcare algorithms. *Nature*.

[Lee, 2017] Lee, E. (2017). APX-hardness of maximizing Nash social welfare with indivisible items. *Information Processing Letters*, 122:17–20.

[Lenstra et al., 1990] Lenstra, J. K., Shmoys, D. B., and Tardos, É. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46:259–271.

[Leskovec et al., 2014] Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014). *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press.

[Li et al., 2021] Li, B., Li, Y., and Wu, X. (2021). Almost proportional allocations for indivisible chores. *CoRR, abs/2103.11849*.

[Lin et al., 2018] Lin, W., Chen, J., Ranjan, R., Bansal, A., Sankaranarayanan, S., Castillo, C. D., and Chellappa, R. (2018). Proximity-aware hierarchical clustering of unconstrained faces. *Image Vis. Comput.*, 77:33–44.

[Lipton et al., 2004] Lipton, R. J., Markakis, E., Mossel, E., and Saberi, A. (2004). On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 125–131.

[Liu et al., 2021] Liu, F.-H., Liu, H.-H., and Wong, P. W. (2021). Greedy is optimal for online restricted assignment and smart grid scheduling for unit size jobs. *Theory of Computing Systems*, 65(6):1009–1032.

[Mahara, 2021] Mahara, R. (2021). Extension of additive valuations to general valuations on the existence of EFX. In *ESA*, volume 204 of *LIPIcs*, pages 66:1–66:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[Mahdian and Yan, 2011] Mahdian, M. and Yan, Q. (2011). Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 597–606.

[Mann et al., 2008] Mann, C. F., Matula, D. W., and Olinick, E. V. (2008). The use of sparsest cuts to reveal the hierarchical community structure of social networks. *Social Networks*, 30(3):223–234.

[Markakis, 2017] Markakis, E. (2017). Approximation algorithms and hardness results for fair division with indivisible goods. *Trends in Computational Social Choice*, pages 231–247.

[Mehta et al., 2013] Mehta, A. et al. (2013). Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368.

[Mehta et al., 2007] Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. (2007). Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22–es.

[Mertzanidis et al., 2024] Mertzanidis, M., Psomas, A., and Verma, P. (2024). Automating food drop: The power of two choices for dynamic and fair food allocation. *arXiv preprint arXiv:2406.06363*.

[Molinaro, 2017] Molinaro, M. (2017). Online and random-order load balancing simultaneously. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1638–1650. SIAM.

[Molinaro and Ravi, 2014] Molinaro, M. and Ravi, R. (2014). The geometry of online packing linear programs. *Mathematics of Operations Research*, 39(1):46–59.

[Moseley and Wang, 2017] Moseley, B. and Wang, J. (2017). Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In *NIPS*, pages 3094–3103.

[Moulin, 2004] Moulin, H. (2004). *Fair division and collective welfare*. MIT press.

[Moulin, 2019] Moulin, H. (2019). Fair division in the internet age. *Annual Review of Economics*, 11(1):407–441.

[Munguía-López and Ponce-Ortega, 2021] Munguía-López, A. d. C. and Ponce-Ortega, J. M. (2021). Fair allocation of potential covid-19 vaccines using an optimization-based strategy. *Process Integration and Optimization for Sustainability*, 5(1):3–12.

[Nielsen and Sun, 2016] Nielsen, F. and Sun, K. (2016). Guaranteed bounds on information-theoretic measures of univariate mixtures using piecewise log-sum-exp inequalities. *Entropy*, 18(12):442.

[Pan et al., 2016] Pan, T., Lo, L., Yeh, C., Li, J., Liu, H., and Hu, M. (2016). Real-time sign language recognition in complex background scene based on a hierarchical clustering classification method. In *IEEE Second International Conference on Multimedia Big Data*, pages 64–67. IEEE Computer Society.

[Persad et al., 2009] Persad, G., Wertheimer, A., and Emanuel, E. J. (2009). Principles for allocation of scarce medical interventions. *The lancet*, 373(9661):423–431.

[Peterson and Su, 2002] Peterson, E. and Su, F. (2002). Four-person envy-free chore division. *Mathematics Magazine*, 75:117 – 122.

[Peterson and Su, 2009] Peterson, E. and Su, F. (2009). N-person envy-free chore division. *arXiv: Combinatorics*.

[Pikhurko, 2000] Pikhurko, O. (2000). On envy-free cake division. *The American Mathematical Monthly*, 107:736 – 738.

[Plaut and Roughgarden, 2020] Plaut, B. and Roughgarden, T. (2020). Almost envy-freeness with general valuations. *SIAM J. Discret. Math.*, 34(2):1039–1068.

[Pratt and Zeckhauser, 1990] Pratt, J. W. and Zeckhauser, R. J. (1990). The fair and efficient division of the Winsor family silver. *Management Science*, 36(11):1293–1301.

[Procaccia, 2009] Procaccia, A. D. (2009). Thou shalt covet thy neighbor's cake. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, page 239–244, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Procaccia, 2020] Procaccia, A. D. (2020). Technical perspective: An answer to fair division's most enigmatic question. *Commun. ACM*, 63(4):118.

[Ramanath et al., 2013] Ramanath, R., Choudhury, M., and Bali, K. (2013). Entailment: An effective metric for comparing and evaluating hierarchical and non-hierarchical annotation schemes. In Dipper, S., Liakata, M., and Pareja-Lora, A., editors, *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*, pages 42–50. The Association for Computer Linguistics.

[Ramezani and Endriss, 2009] Ramezani, S. and Endriss, U. (2009). Nash social welfare in multiagent resource allocation. In *International Workshop on Agent-Mediated Electronic Commerce*, pages 117–131. Springer.

[Rieke and Bogen, 2018] Rieke, A. and Bogen, M. (2018). Help wanted: An examination of hiring algorithms, equity, and bias. *Upturn*.

[Robertson and Webb, 1998] Robertson, J. and Webb, W. (1998). Cake-cutting algorithms - be fair if you can.

[Rösner and Schmidt, 2018] Rösner, C. and Schmidt, M. (2018). Privacy preserving clustering with constraints. In *ICALP*, pages 96:1–96:14.

[Roy and Pokutta, 2016] Roy, A. and Pokutta, S. (2016). Hierarchical clustering via spreading metrics. In *NIPS*, pages 2316–2324.

[Seaver, 2017] Seaver, N. (2017). Algorithms as culture: Some tactics for the ethnography of algorithmic systems. *Big data & society*, 4(2):2053951717738104.

[Segal-Halevi and Suksompong, 2020] Segal-Halevi, E. and Suksompong, W. (2020). How to cut a cake fairly: A generalization to groups. *The American Mathematical Monthly*, 128(1):79–83.

[Seiden, 2002] Seiden, S. S. (2002). On the online bin packing problem. *Journal of the ACM (JACM)*, 49(5):640–671.

[Selvan et al., 2005] Selvan, A., Cole, L., Spackman, L., Naylor, S., and C, W. (2005). Hierarchical cluster analysis to aid diagnostic image data visualization of ms and other medical imaging modalities. In *Molecular Biotechnology*.

[Slaughter et al., 2020] Slaughter, R. K., Kopec, J., and Batal, M. (2020). Algorithms and economic justice: A taxonomy of harms and a path forward for the federal trade commission. *Yale JL & Tech.*, 23:1.

[Springer et al., 2024] Springer, M., Hajiaghayi, M., and Yami, H. (2024). Almost envy-free allocations of indivisible goods or chores with entitlements. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9901–9908.

[Steinhaus, 1948] Steinhaus, H. (1948). The problem of fair division. *Econometrica*, 16:101–104.

[Stern and Gamow, 1958] Stern, M. and Gamow, G. (1958). Puzzle-math.

[Sun et al., 2021] Sun, A., Chen, B., and Doan, X. V. (2021). Connections between fairness criteria and efficiency for allocating indivisible chores. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, pages 1281–1289. ACM.

[Sweeney, 2013] Sweeney, L. (2013). Discrimination in online ad delivery. *ACM Queue*.

[Tan and Cao, 2006] Tan, Z. and Cao, S. (2006). Semi-online machine covering on two uniform machines with known total size. *Computing*, 78(4):369–378.

[Taylor, 2024] Taylor, P. (2024). Data growth worldwide 2010-2028 — statista.

[Varian, 1973] Varian, H. R. (1973). Equity, envy, and efficiency.

[Vossen, 2002] Vossen, T. W. (2002). *Fair allocation concepts in air traffic management*. PhD thesis, University of Maryland, College Park.

[Walsh, 2011] Walsh, T. (2011). Online cake cutting. In *Algorithmic Decision Theory: Second International Conference, ADT 2011, Piscataway, NJ, USA, October 26-28, 2011. Proceedings 2*, pages 292–305. Springer.

[Woeginger, 1997] Woeginger, G. J. (1997). A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154.

[Wu et al., 2021] Wu, X., Li, B., and Gan, J. (2021). Budget-feasible maximum Nash social welfare is almost envy-free. International Joint Conferences on Artificial Intelligence Organization.

[Wu et al., 2023] Wu, X., Zhang, C., and Zhou, S. (2023). Weighted EF1 allocations for indivisible chores.

[Wu et al., 2014] Wu, Y., Cheng, T., and Ji, M. (2014). Optimal algorithms for semi-online machine covering on two hierarchical machines. *Theoretical Computer Science*, 531:37–46.

[Yokoyama and Igarashi, 2023] Yokoyama, T. and Igarashi, A. (2023). Asymptotic existence of class envy-free matchings.

[Yuen and Suksompong, 2023] Yuen, S. M. and Suksompong, W. (2023). Extending the characterization of maximum Nash welfare. *Economics Letters*, 224:111030.

[Zeng and Psomas, 2020] Zeng, D. and Psomas, A. (2020). Fairness-efficiency tradeoffs in dynamic fair division. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 911–912.

[Zhou et al., 2008] Zhou, Y., Chakrabarty, D., and Lukose, R. (2008). Budget constrained bidding in keyword auctions and online knapsack problems. In *International Workshop on Internet and Network Economics*, pages 566–576. Springer.