Foundations of computing Volume 15

Andrzej Kaczmarczyk

Algorithmic Aspects of Resource Allocation and Multiwinner Voting: Theory and Experiments

Universitätsverlag der TU Berlin

Andrzej Kaczmarczyk

Algorithmic Aspects of Resource Allocation and Multiwinner Voting: Theory and Experiments

The scientific series *Foundations of computing* of the Technische Universität Berlin is edited by:
Prof. Dr. Stephan Kreutzer
Prof. Dr. Uwe Nestmann
Prof. Dr. Rolf Niedermeier

Foundations of computing | 15

Andrzej Kaczmarczyk

Algorithmic Aspects of Resource Allocation and Multiwinner Voting: Theory and Experiments

Universitätsverlag der TU Berlin

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the internet at http://dnb.dnb.de.

Universitätsverlag der TU Berlin, 2021
http://www.verlag.tu-berlin.de

Fasanenstr. 88, 10623 Berlin
Tel.: +49 (0)30 314 76131 / Fax: -76133
E-Mail: publikationen@ub.tu-berlin.de

Zugl.: Berlin, Techn. Univ., Diss., 2020
Gutachter: Prof. Dr. Rolf Niedermeier (TU Berlin)
Gutachter: Prof. Dr. Sylvain Bouveret (Université Grenoble Alpes)
Gutachter: Prof. Dr. Felix Brandt (TU München)
Die Arbeit wurde am 10. Dezember 2020 an der Fakultät IV unter Vorsitz von Prof. Dr. Markus Brill erfolgreich verteidigt.

Cover image: *Mount Kazbek* by Andrzej Kaczmarczyk, 2012
CC BY 4.0 | https://creativecommons.org/licenses/by/4.0

Print: docupoint GmbH
Layout/Typesetting: Andrzej Kaczmarczyk

ORCID iD Andrzej Kaczmarczyk: 0000-0003-1401-0157
http://orcid.org/0000-0003-1401-0157

ISBN 978-3-7983-3215-7 (print)
ISBN 978-3-7983-3216-4 (online)

ISSN 2199-5249 (print)
ISSN 2199-5257 (online)

Published online on the institutional repository of the Technische Universität Berlin:
DOI 10.14279/depositonce-12056
http://dx.doi.org/10.14279/depositonce-12056

Zusammenfassung

Diese Arbeit befasst sich mit der Untersuchung von Themen des Forschungsgebiets Computational Social Choice im Lichte realer Anwendungen. Dabei trägt sie zu einem besseren Verständnis der Bereiche der fairen Zuordnung und der Mehrgewinnerwahlen bei. Für beide Konzepte schlagen wir – inspiriert von realen Anwendungen – verschiedene neue Begriffe und Erweiterungen bestehender Modelle vor. Anschließend analysieren wir die Komplexität der Beantwortung von Berechnungsfragen, die durch die eingeführten Konzepte aufgeworfen werden. Dabei fokussieren wir uns auf die parametrisierte Komplexität. Hierzu identifizieren wir verschiedene Parameter, welche natürliche Merkmale der von uns untersuchten Berechnungsprobleme beschreiben. Durch die Nutzung dieser Parameter entwickeln wir erfolgreich effiziente Algorithmen für Spezialfälle der untersuchten Probleme. Wir ergänzen unsere Analyse indem wir zeigen, welche Parameter vermutlich nicht verwendet werden können um effiziente Algorithmen zu finden. Dabei zeichnen wir ein umfassendes Bild der Berechnungskomplexität der untersuchten Probleme. Insbesondere konzentrieren wir uns auf vier Themen, die wir, gruppiert nach unseren beiden Schwerpunkten, unten vorstellen. Für alle Themen bis auf eines präsentieren wir Experimente, die auf Implementierungen der von uns neu entwickelten Algorithmen basieren.

Wir konzentrieren uns zunächst auf die faire Zuordnung unteilbarer Ressourcen. Hier betrachten wir eine Menge unteilbarer Ressourcen und eine Gruppe von Agenten. Jeder Agent gibt eine Bewertung des Nutzens jeder Ressource ab und die Aufgabe besteht darin, eine "‘faire"’ Zuordnung der Ressourcen zu finden, wobei jede Ressource höchstens einem Agenten zugeordnet werden kann. Innerhalb dieses Bereiches konzentrieren wir uns auf die beiden folgenden Problemstellungen.

Der soziale Kontext bei der fairen Zuordnung unteilbarer Ressourcen. In vielen Szenarien, in denen Ressourcen zugeordnet werden sollen, ist es unwahrscheinlich, dass jeder Agent alle anderen kennt. Vorstellbar ist beispielsweise ein Szenario, in dem die Agenten Mitarbeiter eines großen Unternehmens repräsentieren. Es ist höchst unwahrscheinlich, dass jeder Mitarbeiter jeden anderen Mitarbeiter kennt. Motiviert durch solche Szenarien entwickeln wir ein neues Modell der graph-basierten Neidfreiheit. Wir erweitern den klassischen Neidfreiheitsbegriff um die sozialen Beziehungen von Agenten, die durch soziale Netzwerke modelliert

werden. Einerseits zeigen wir, dass wenn das soziale Netzwerk der Agenten einfach ist (zum Beispiel, wenn es sich um einen gerichteten azyklischen Graph handelt), in manchen Fällen faire Zuordnungen effizient gefunden werden können. Andererseits stellen wir diesen algorithmisch positiven Ergebnissen mehrere NP-schweren Fällen entgegen. Ein Beispiel für einen solchen Fall sind soziale Netzwerke mit einem konstanten Knotengrad.

Faire Zuteilung an wenige Agenten mit begrenzter Rationalität. Begrenzte Rationalität beschreibt die Idee, dass Menschen aufgrund kognitiver Grenzen dazu neigen, Probleme, mit denen sie konfrontiert werden, zu vereinfachen. Eine mögliche Folge dieser Grenzen ist, dass menschliche Agenten in der Regel einfache Bewertungen der gewünschten Ressourcen abgeben; beispielsweise könnten Agenten die verfügbaren Ressourcen nur in zwei Gruppen, erwünschte und unerwünschte Ressourcen, kategorisieren. Durch Anwendung von Techniken zum Lösen von Ganzzahligen Linearen Programmen zeigen wir, dass unter der Annahme einer kleinen Anzahl von Agenten die Ausnutzung begrenzter Rationalität dabei hilft, effiziente Algorithmen zum Finden neidfreier und Pareto-effizienter Zuweisungen zu entwickeln. Weiterhin zeigen wir, dass unser Ergebnis ein allgemeines Verfahren liefert, welches auf eine Reihe verschiedener Fairnesskonzepte angewendet werden kann, wie zum Beispiel Neidfreiheit bis auf ein Gut oder Neidfreiheit bis auf irgendein Gut. Auf diese Weise gewinnen wir effiziente Algorithmen für eine Reihe fairer Zuordnungsprobleme (wenige Agenten mit begrenzter Rationalität vorausgesetzt). Darüber hinaus zeigen wir empirisch, dass unsere Technik in der Praxis anwendbar ist.

Weiterhin untersuchen wir Mehrgewinnerwahlen, bei denen uns eine Menge von Wählern sowie ihre Präferenzen über eine Reihe von Kandidaten gegeben sind. Das Ergebnis eines Mehrgewinnerwahlverfahrens ist eine Gruppe (oder eine Menge von Gruppen im Falle eines Unentschiedens) von Kandidaten, welche die Präferenzen der Wähler am besten einem bestimmten Ziel folgend widerspiegeln. In diesem Kontext untersuchen wir die folgenden Themen.

Die Robustheit von Wahlergebnissen. Wir untersuchen, wie robust die Ergebnisse von Mehrgewinnerwahlen gegenüber möglicher Fehler der Wähler sind. Unter der Annahme, dass jeder Wähler eine Stimme in Form einer Rangliste von Kandidaten abgibt, modellieren wir einen Fehler als einen Tausch benachbarter Kandidaten in der Rangliste. Wir zeigen, dass für Wahlregeln wie SNTV, k-Approval und k-Borda die minimale Anzahl an Vertauschungen, welche zu einer Ergebnisänderung führt, einfach zu berechnen ist. Für STV und die Chamberlin-Courant-Regel ist diese Aufgabe allerdings NP-schwer. Wir schließen unsere Untersuchung der Robustheit unterschiedlicher Wahlregeln ab mit einer experi-

mentellen Evaluierung der durchschnittlichen Anzahl zufälliger Vertauschungen, die zu einer Änderung des Ergebnisses führen.

Strategische Abstimmung bei Wahlen mit mehreren Gewinnern. Wir fragen, ob eine bestimmte Gruppe von kooperierenden Wählern ein Wahlergebnis zu ihren Gunsten manipulieren kann. Dabei konzentrieren wir uns auf die k-Approval-Wahlregel. Wir zeigen, dass die Berechnungskomplexität der besagten Manipulation eine reiche Struktur besitzt. Auf der einen Seite identifizieren wir mehrere Fälle in denen das Problem in Polynomzeit lösbar ist. Auf der anderen Seite identifizieren wir jedoch auch NP-schwere Fälle. Für einige von ihnen zeigen wir, wie die Berechnungsschwere durch parametrisierte Algorithmen umgangen werden kann. Wir präsentieren zudem experimentelle Untersuchungen, welche darauf hindeuten, dass unsere Algorithmen in der Praxis anwendbar sind.

Abstract

This thesis is concerned with investigating elements of computational social choice in the light of real-world applications. We contribute to a better understanding of the areas of fair allocation and multiwinner voting. For both areas, inspired by real-world scenarios, we propose several new notions and extensions of existing models. Then, we analyze the complexity of answering the computational questions raised by the introduced concepts. To this end, we look through the lens of parameterized complexity. We identify different parameters which describe natural features specific to the computational problems we investigate. Exploiting the parameters, we successfully develop efficient algorithms for specific cases of the studied problems. We complement our analysis by showing which parameters presumably cannot be utilized for seeking efficient algorithms. Thereby, we provide comprehensive pictures of the computational complexity of the studied problems. Specifically, we concentrate on four topics that we present below, grouped by our two areas of interest. For all but one topic, we present experimental studies based on implementations of newly developed algorithms.

We first focus on fair allocation of indivisible resources. In this setting, we consider a collection of indivisible resources and a group of agents. Each agent reports its utility evaluation of every resource and the task is to "fairly" allocate the resources such that each resource is allocated to at most one agent. We concentrate on the two following issues regarding this scenario.

The social context in fair allocation of indivisible resources. In many fair allocation settings, it is unlikely that every agent knows all other agents. For example, consider a scenario where the agents represent employees of a large corporation. It is highly unlikely that every employee knows every other employee. Motivated by such settings, we come up with a new model of graph envy-freeness by adapting the classical envy-freeness notion to account for social relations of agents modeled as social networks. We show that if the given social network of agents is simple (for example, if it is a directed acyclic graph), then indeed we can sometimes find fair allocations efficiently. However, we contrast tractability results with showing NP-hardness for several cases, including those in which the given social network has a constant degree.

Fair allocations among few agents with bounded rationality. Bounded rationality is the idea that humans, due to cognitive limitations, tend to simplify problems that they face. One of its emanations is that human agents usually

tend to report simple utilities over the resources that they want to allocate; for example, agents may categorize the available resources only into two groups of desirable and undesirable ones. Applying techniques for solving integer linear programs, we show that exploiting bounded rationality leads to efficient algorithms for finding envy-free and Pareto-efficient allocations, assuming a small number of agents. Further, we demonstrate that our result actually forms a framework that can be applied to a number of different fairness concepts like envy-freeness up to one good or envy-freeness up to any good. This way, we obtain efficient algorithms for a number of fair allocation problems (assuming few agents with bounded rationality). We also empirically show that our technique is applicable in practice.

Further, we study multiwinner voting, where we are given a collection of voters and their preferences over a set of candidates. The outcome of a multiwinner voting rule is a group (or a set of groups in case of ties) of candidates that reflect the voters' preferences best according to some objective. In this context, we investigate the following themes.

The robustness of election outcomes. We study how robust outcomes of multiwinner elections are against possible mistakes made by voters. Assuming that each voter casts a ballot in a form of a ranking of candidates, we represent a mistake by a swap of adjacent candidates in a ballot. We find that for rules such as SNTV, k-Approval, and k-Borda, it is computationally easy to find the minimum number of swaps resulting in a change of an outcome. This task is, however, NP-hard for STV and the Chamberlin-Courant rule. We conclude our study of robustness with experimentally studying the average number of random swaps leading to a change of an outcome for several rules.

Strategic voting in multiwinner elections. We ask whether a given group of cooperating voters can manipulate an election outcome in a favorable way. We focus on the k-Approval voting rule and we show that the computational complexity of answering the posed question has a rich structure. We spot several cases for which our problem is polynomial-time solvable. However, we also identify NP-hard cases. For several of them, we show how to circumvent the hardness by fixed-parameter tractability. We also present experimental studies indicating that our algorithms are applicable in practice.

Preface

The thesis contains a collection of outcomes of my research activity at TU Berlin in the Algorithmics and Computational Complexity group of Prof. Dr. Rolf Niedermeier from November 2016 to July 2020. During the whole period, my research activities were funded by Deutsche Forschungsgemeinschaft (DFG) as a part of the project *AFFA: Algorithms for Fair Allocations (NI 369/15 and BR 5207/1)*. This research project was jointly led by Robert Bredereck and Rolf Niedermeier.

The presented results are mainly based on conference publications prepared with close collaboration with several coauthors, who are, in alphabetical order, Robert Bredereck, Piotr Faliszewski (AGH University of Science and Technology, Cracow, Poland), Dušan Knop (Czech Technical University in Prague, Prague, the Czech Republic), Rolf Niedermeier, Piotr Skowron (University of Warsaw, Warsaw, Poland), and Nimrod Talmon (Ben-Gurion University of the Negev, Beer-Sheva, Israel). Furthermore, some experiments were conducted on software implemented by Lydia Kalkbrenner (as a part of her bachelor's thesis supervised by Rolf Niedermeier with support from Robert Bredereck and myself) and by Aleksander Figiel during his work as a student assistant in the Algorithmics and Computational Complexity group.

Below, I briefly list my contributions to the publications laying the foundation for the respective chapters.

Chapter 4 The idea of augmenting the standard notion of envy-freeness with a social network over agents (thus relaxing the standard envy-freeness) appeared during brainstorming at one of the early meetings regarding the AFFA project. In the conference version, which I presented at the *17th International Conference on Autonomous Agents and Multiagents Systems (AAMAS '18)* [BKN18] and later (as a poster) at *DIMEA Days 2019* in Brno, the majority of the results was jointly developed by all authors. Then, supported by Robert Bredereck, I was mainly responsible for preparing a journal version, extending the conference paper by parameterized complexity results. A revision of the journal version is currently being prepared for the second round of reviewing for the journal *Artificial Intelligence*.

Chapter 5 The crucial idea behind the main result, conjectured by Robert Bredereck, was shared with Dušan Knop and me. Shortly later, Dušan provided the technical result implementing Robert's idea. The result was then wrote up, revised and polished by all authors who jointly prepared a paper later presented by me at the *20th ACM Conference on Economics and Computation (ACM EC '19)* [Bre+19b]. Then, I extended the model and together with Dušan Knop we revised some of the results (regarding different relaxations of envy-freeness) adapting them to the new model. Aleksander Figiel, guided by Dušan Knop, Robert Bredereck and myself, implemented the algorithms from the conference paper. Thereby, he developed a framework for testing the running times and the outcomes of the algorithms which I used used to conduct the experiments in the thesis.

Chapter 7 The problem of analyzing the robustness of multiwinner voting rules was proposed to Robert Bredereck, Rolf Niedermeier, Piotr Skowron, Nimrod Talmon, and myself by Piotr Faliszewski during his stay at TU Berlin with the group. The theoretical results, developed jointly, were augmented with an experimental study, conducted by me closely collaborating with Piotr Faliszewski. Robert Bredereck presented the work, whose write-up was jointly prepared by all authors, at the *10th International Symposium on Algorithmic Game Theory (SAGT '17)* [Bre+17]. The long version of the paper, improved by Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier and myself in both theoretical and practical parts, has been published in the journal *Artificial Intelligence* [Bre+21a].

Chapter 8 Studying coalitional manipulation, later narrowed down to coalitional manipulation in shortlisting scenarios, was proposed by Robert Bredereck. I conceived the ideas for polynomial-time algorithms which were then checked, wrote up, and proofread by all authors. All other parts of the resulting paper, presented by me at *26th International Joint Conference on Artificial Intelligence (IJCAI '17)* [BKN17], were developed jointly by all authors. I was mainly responsible for preparing an extended version of the paper for a journal submission. The extended version has recently been published in the journal *Autonomous Agents and Multi-Agent Systems* [BKN21]. Robert Bredereck and I were helping Rolf Niedermeier supervising Lydia Kalkbrenner who implemented several algorithms from the conference paper as a part of her bachelor's thesis. After few fixes, I used these implementations to conduct experiments.

In addition to the above-mentioned works, I was involved in other research projects that are not covered by my thesis. I contributed to a theoretical study on improving the results from Chapter 5 [Bre+20c] as well as to an experimental study focused on applying these results in practice [Bre+21b], an experimental study on electing committees representing voters proportionally [Bre+19a], a theoretical study of strategic voting in single-winner elections [KF19], a theoretical and practical study on strategic voting in apportionment elections [Bre+20a], a theoretical analysis of the problem of selecting a collective set of items [Bre+20b], a theoretical and practical analysis of parallel elections [Boe+20], a theoretical analysis of scheduling problems in the context of resources [Ben+21], and two works concerning selecting multiple committees [BFK20, BKN20].

Acknowledgements I am very grateful to my supervisor, Rolf Niedermeier, who made my journey through carrying out the research and writing the thesis possible. His patience and eagerness to help had been constantly motivating me, while his deep expertise, insightful feedback and keen ideas pushed me to lift my work to a higher level. I am also very thankful to Sylvain Bouveret and Felix Brandt, the external reviewers of my thesis, whose valuable comments allowed me to improve the presentation of my thesis. I am also grateful to the Deutsche Forschungsgemeinschaft for financially supporting me during my research.

I would like to acknowledge all my coauthors for thousands of inspiring and fruitful discussions, numerous tedious proofreading rounds, and lots of advice. Next to my closest collaborator, Robert Bredereck, from whom I have learned the most, I want to thank (in alphabetical order) Matthias Bentert, Niclas Boehmer, Piotr Faliszewski, Aleksander Figiel, Till Fluschnik, Michał Furdyna, Péter Györgyi, Dušan Knop, Martin Lackner, Piotr Skowron, and Nimrod Talmon. The time I spent at TU Berlin would have never been so memorable without other (former) group members and friends of the group with whom I am not co-authoring any paper (I wish we could change that one day!). In alphabetical order, these are: Markus Brill, Jiehua Chen (Thank you very much for the dissertation latex template!), Vincent Froese, Anne-Marie George, Klaus Heeger, Danny Hermelin, Anne-Sophie Himmel, Junjie Luo, Leon Kellerhals, Tomohiro Koana, Christian Komusiewicz, George B. Mertzios, André Nichterlein, Malte Renken, Ulrike Schmidt-Kraepelin, Manuel Sorge, Ondřej Suchý, Mathias Weller, and Philipp Zschoche. Special thanks go to Christlinde Thielcke for solicitously and tirelessly helping me with all administrative matters.

Undoubtedly, the thesis would not exist without the support of my family,

especially my grandparents Gertruda and Czesław, my parents Ruta and Marek, my brother Jacek and my dear wife Marta. Their care, inspiration, and good words kept me motivated and focused on the goal while I was going through occasional hard times.

Last but not least, *soli Deo gloria*.

Contents

1. Introduction **1**

2. Preliminaries and Notation **5**
2.1. Basics 5
2.2. Vectors and Matrices 5
2.3. Graph Theory 6
2.4. Computational Complexity 6
2.5. Parameterized Computational Complexity 8
2.6. Integer Linear Programming 10
2.7. Experimental Environment 11

I. Resource Allocation. Dealing with Private Bundles 13

3. Formalism of Resource Allocations **15**

4. Graph Envy-Freeness **17**
4.1. Introduction 17
4.2. Basic Definitions 23
4.3. Model and Discussion 25
4.4. Finding Weakly Graph-Envy-Free Allocations 31
4.5. Finding Strongly Graph-Envy-Free Allocations 50
4.6. Conclusion 62

5. High Multiplicity Allocations **67**
5.1. Introduction 67
5.2. Preliminaries 71
5.3. Seeking Envy-Free Pareto-Efficient Allocations 78
5.4. Beyond Envy-Freeness 89
5.5. Experimental Evaluation 96
5.6. Conclusions 101

II. Multiwinner Voting. Dealing with Collective Sets of Resources 105

6. Elections, Multiwinner Voting Rules, and Election Generation 107
6.1. Elections and Multiwinner Voting 107
6.2. Generating Synthetic Election Data 111

7. Robustness of Multiwinner Voting Rules 113
7.1. Introduction 113
7.2. Preliminaries 115
7.3. Classical Computational Complexity 123
7.4. Parameterized Computational Complexity 137
7.5. Beyond the Worst Case: An Experimental Evaluation 146
7.6. Conclusions 150

8. Coalitional Manipulation for Multiwinner Elections 153
8.1. Introduction 153
8.2. Preliminaries 158
8.3. Complexity of Tie-Breaking 171
8.4. Complexity of Coalitional Manipulation 177
8.5. Experimental Insights 198
8.6. Conclusion 203

9. Conclusion 207
9.1. Catching up with the Future 209
9.2. Epilogue 212

Bibliography 213

CHAPTER 1

Introduction

This work provides a study of computational aspects of two prominent fields of social choice theory: fair allocation of indivisible resources and multiwinner voting. For both areas of our interest, we conduct theoretical analyses and then apply (elements of) our findings in experimental studies.

In a nutshell, the problem of fair allocation of indivisible resources, which we study in Part I, is to distribute a collection of unsplittable resources to a set of agents. Each agent is endowed with its private valuations of the resources and each resource can be allocated to only one agent. The goal is to allocate resources in the fairest possible manner. Arriving at a better understanding of the meaning of "the fairest" is one of the objectives of this work. Typical real-world examples of fair allocation include dividing inheritance [PZ90], spectrum (frequency) allocation [Guo+16], distributing chores [Mar02], and allocating resources in virtualized computing environments [Sti+10].

At a high level, multiwinner voting, covered in Part II, is a procedure leading to choosing (from a usually large pool of candidates) a fixed number of candidates that are most suitable (with respect to some criterion such as excellence) according to the preferences of the voters. An archetypical application is selecting (various types of) governments or supervisory boards. Besides politics, multiwinner voting also naturally appears, for example, when shortlisting nominees for various prizes such as the Grammy Awards [Rec20], in recommender systems [Cha+19], or in deciding on a funding for research projects [Min19].

Our goals for both areas are threefold. First, we aim at bringing them closer to reality by proposing extensions of the previously known concepts or asking new, practically relevant questions. Second, we seek efficient algorithms to address our newly defined problems and extensions in practice. Thus we aim at providing algorithmic tools for a further analysis. As a necessary step to achieve the second goal, we also categorize the introduced problems (or problem variants) with respect to their computational hardness charting their borders of tractability. Third, we perform experimental studies using our findings and observations from the first two mentioned points.

The thesis consists of four main chapters (two chapters per studied area).

In each of them, we analyze one topic from the respective area and carry out a study aiming at achieving the goals described in the preceding paragraph. Below, we describe the problems we focus on in the thesis in more detail.

- In Chapter 4, we incorporate social relations into the traditional problem of fair allocation. Thus we aim at reflecting the phenomenon that humans tend to pay greater attention to their "socially-close" peers than to "socially-distant" individuals.
- In Chapter 5, we introduce a variant of the fair allocation of indivisible resources problem which is suitable for scenarios involving resources that come in many indistinguishable copies. For example, this applies for food banks where there could be a lot of packages of rice, pasta, and the like to distribute.
- In Chapter 7, we pose and address a new question related to multiwinner elections: "How robust are the outcomes of different multiwinner voting rules against unintentional mistakes of the voters?". Answering this question for a *single* election can in practice serve as a measure of confidence for an election outcome. For example, if the election models a panel of experts selecting projects to be funded, then a low confidence of the experts for the chosen projects could indicate a need to hire more experts.
- In Chapter 8, we introduce the concept of coalitional manipulation into the domain of multiwinner voting rules and obtain a fairly general model of coalitional manipulation in multiwinner elections. The model opens up avenues for a more fine-grained investigation on the practical vulnerability of an election to a targeted and coordinated manipulative action. In particular, the model could be of use for an interdisciplinary behavioral study on the phenomenon of manipulation (similar to that of Scheuerman et al. [Sch+19]).

We contribute to a better understanding of the above-mentioned problems providing several efficient (polynomial-time) algorithms. Whenever this is impossible, then we successfully use the toolbox of parameterized complexity to get round computational hardness. Let us briefly explain the intuition behind the concept of parameterized complexity. The idea is to describe certain features of a problem by a so-called parameter, usually being an integer. Then, after identifying a parameter of interest that represents a studied feature well (which is, per se, also a nontrivial task), the next step is to exploit the additional

information that the parameter provides. The hope is to come up with efficient algorithmic solutions tailored to the considered parameter. In our work, we study several different natural parameterizations for which we used various approaches such as dynamic programming, special cases of efficiently-solvable Integer Linear Programming formulations, or data reduction (implicitly). We obtain several efficient algorithms for the corresponding (constrained) variants of the considered problems. Exploiting the concept of parameterized reductions, we also provide numerous parameterized intractability results. These intuitively show that for several parameters of the considered problems one presumably cannot exploit the parameter to achieve an efficient algorithm.

Finally, in all but one of the four aforementioned chapters, we conduct experimental studies. In Chapters 5 and 8, we empirically show that the algorithms proposed therein are indeed applicable in practice (occasionally, theoretically efficient algorithms turn out to be practically computationally infeasible). To this end, we conducted experiments on real-world data (from spliddit.org [Pro+20], provided to us by the authors of this web service [GP15]), as well as on synthetic data generated using well-established models. In Chapter 7, we empirically draw qualitative conclusions about the behavior of different multiwinner voting rules for various elections. In this study we included real-world data from the Preflib library [MW13] as well as synthetic data. We do not present experiments for the problems considered in Chapter 4 because of a lack of suitable data and reliable and meaningful synthetic models.

We take a moment to discuss the relations between fair allocation of indivisible resources and multiwinner voting. At first sight, these two areas might appear very obliquely related, especially considering that in the fair allocation domain we speak of resources and agents, whereas in multiwinner voting we deal with voters and candidates. However, observe that both voters and agents play similar roles in the two areas. First, they express the preferences over possible choices (either resources or candidates). Second, they are focal points of both scenarios: we either want to guarantee fairness for the agents or to aggregate preferences of the voters in the best possible way. From this perspective, both resources and candidates are, in a sense, subject to the agents or voters. Altogether, these similarities justify treating voters as agents and candidates as resources. As a result, we can describe multiwinner voting in the domain of fair allocation of indivisible resources as follows. In multiwinner voting, we select one collective set of resources that are, on an intuitive level, shared by all agents equally. Naturally, the selected set of resources should fulfill a predefined goal based on agents preferences as much as possible. Additionally, the selected set of resources

could be subject to further constraints like the number of resources it contains, a cost of resources, or similar issues. The presented interpretation has already been adopted by several works in the domain of multiwinner voting [Bre+20b, SFL16]. It has also been used in the setting called participatory budgeting (see a survey by Aziz and Shah [AS20]). In participatory budgeting each candidate comes at a cost and the selected set of candidates cannot exceed a given budget.

Each of the two parts of the thesis consists of three chapters: one short preliminary chapter devoted to introducing necessary definitions and two chapters describing our results. Additionally every part starts with a brief description of the respective area. We conclude the thesis in Chapter 9.

CHAPTER 2

Preliminaries and Notation

In this chapter, we provide basic notation and introduce basic concepts used in the thesis.

2.1 Basics

We use the standard brace notation for sets and their elements. We also use the brace notation for collections, sometimes also called multisets. Thus, we explicitly say whether we mean a set or a collection whenever skipping this comment would lead to ambiguities. For some positive integer n, we denote the set $\{1, 2, \ldots, n\}$ by $[n]$. Let A and B be sets. By $|A|$ we denote the number of elements in A, by $\binom{A}{2}$ we denote all size-two subsets of A, and by 2^A we mean all subsets of A. By $A \times B \coloneqq \{(a, b) \colon a \in A \wedge b \in B\}$ we denote the *Cartesian product* of A and B.

2.2 Vectors and Matrices

We use boldface letters for vectors and normal letters for their entries. For example, for some positive integer y, $\mathbf{x} = (x_1, x_2, \ldots, x_y)$ is a vector of dimension y. Whenever we deal with matrices, we always refer to them by single capital letters. For a matrix with r rows and c columns, we say that it is of dimension $r \times c$. Symbolically, we denote a matrix A of dimension $r \times c$ whose entries are integers (an integral matrix) as $A \in \mathbb{Z}^{r \times c}$. All vectors can alternatively be considered as matrices with either of dimensions equal to one. Because we extensively use integer linear programs in which it is convenient to speak about a "solution vector," we follow a convention that each vector is a column vector. This is equivalent to saying that each vector is a single-column matrix. So, a vector $\mathbf{x} = (x_1, x_2, \ldots, x_y)$ is in fact a matrix with dimensions $y \times 1$ and thus the following:

$$\mathbf{x} \equiv \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_y \end{pmatrix}.$$

2.3 Graph Theory

Throughout this section we define standard graph-theoretical concepts used in the thesis.

Undirected Graphs A (simple undirected) *graph* $G = (V, E)$ is a tuple consisting of a set V of *vertices* and a set $E \subseteq \binom{V}{2}$ of *edges*. For an edge $e = \{v, v'\} \in E$, we say that v is *adjacent* to v' and that e is *incident* to v and to v'. We call two adjacent vertices *neighbors*. For every vertex $v \in V$, by $N(v) \coloneqq \{v' \colon \{v, v'\} \in E\}$ we denote the *neighborhood* of v and by $\Delta(v) \coloneqq |N(v)|$ the *degree* of v.

Directed Graphs A (simple) *directed graph* $G = (V, A)$ is a tuple consisting of a set V of *vertices* and a set $A \subseteq \{(v, v') \colon v, v' \in V \wedge v \neq v'\}$ of *arcs*.

For an arc $e = (v, v') \in A$, we say that v and v' are *adjacent* to each other, and that e is *incident* to v and to v'. We call two adjacent vertices *neighbors*. For an arc $e = (v, v') \in A$, we also say that v *points to* v'. We call e an *incoming* arc of v', and an *outgoing* arc of v. For a vertex $v \in V$, we refer to the set $N^+(v) \coloneqq \{v' \colon (v, v') \in A\}$ as the *out-neighborhood* of v and we call $\Delta^+(v) \coloneqq \left|N^+(v)\right|$ the *out-degree* of v. Analogously, we define the *in-neighborhood* of v as $N^-(v) \coloneqq \{v' \colon (v', v) \in A\}$ and the *in-degree* of v as $\Delta^-(v) \coloneqq \left|N^-(v)\right|$.

2.4 Computational Complexity

In this section we give a short primer on computational complexity theory, giving a canonical way of studying the inherent difficulty of computational problems.

At the foundation of computational complexity theory are *decision problems*. Let Σ be a finite alphabet and let Σ^* be the set of all finite words over Σ. Then, given a *language* $\mathcal{L} \in 2^{\Sigma^*}$, the corresponding decision problem is to decide whether a given string $\mathcal{I} \in \Sigma^*$, called an *instance*, belongs to $\mathcal{L}$ or not. The alphabet is usually assumed to be binary (i.e., $\Sigma = \{0, 1\}$). An example of a decision problem is testing whether a given number is prime. Here, assuming the standard binary alphabet Σ, a single instance is a binary representation of a number and the set $\mathcal{L}$ is the set of binary representations of all primes.

The difficulty of a decision problem is most often measured with respect to the so-called *worst-case computational complexity*. The worst-case computational complexity is the worst possible running time needed by any algorithm to decide an instance of the problem; here the running time is an alias for the number

of basic operations in a given computational model. In our work, the only computational model considered is that of *Random Access Machines*.

We say that a problem $\mathcal{L}$ is solvable in *polynomial time* if there exists a *deterministic* Turing machine that, for every instance $\mathcal{I}$, decides if $\mathcal{I}$ belongs to $\mathcal{L}$ in running time $\langle\mathcal{I}\rangle^c$ for some constant c, where $\langle\mathcal{I}\rangle$ is the length of an encoding of the instance. The class P is a collection of all polynomial-time solvable problems, which are considered to be "theoretically tractable." Another important class, NP, consists of all problems that can be solved in polynomial time by a *non-deterministic* Turing machine. It is clear that $\mathsf{P} \subseteq \mathsf{NP}$. However, even though non-deterministic Turing machines seem to be much more powerful than their deterministic counterparts, the question whether $\mathsf{P} = \mathsf{NP}$ remains a long-standing open question in computer science. The answer is, however, strongly conjectured to be negative and we assume so throughout this thesis. This assumption gives birth to a widespread belief in "theoretically intractable" problems—those in $\mathsf{NP} \setminus \mathsf{P}$.

A central tool used in computational complexity analysis is the concept of polynomial-time many-one reductions.

Definition 2.1. Let Σ be a finite alphabet. A *polynomial-time many-one reduction* from a problem $\mathcal{L} \in \Sigma^*$ to a problem $\mathcal{L}' \in \Sigma^*$ is a polynomial-time computable function $f\colon \Sigma^* \to \Sigma^*$ such that, for every instance $\mathcal{I} \in \Sigma^*$, $\mathcal{I} \in \mathcal{L} \Leftrightarrow f(\mathcal{I}) \in \mathcal{L}'$.
We say that a problem $\mathcal{L}$ is (polynomial-time many-one) reducible to a problem $\mathcal{L}'$ if there is a polynomial-time many-one reduction from $\mathcal{L}$ to $\mathcal{L}'$.

Polynomial-time many-one reductions allow to order problems with respect to their hardness. Briefly, if a problem $\mathcal{L}$ is reducible to a problem $\mathcal{L}'$, then, knowing how to solve $\mathcal{L}'$, we can solve every instance of $\mathcal{L}$ by reducing to an equivalent instance of $\mathcal{L}'$ using a polynomial-time reduction. Thus, intuitively, $\mathcal{L}'$ is at least as hard as $\mathcal{L}$. We are now ready to formally define the class of NP-hard problems, that is, all problems at least as hard as every problem in NP, and the class of NP-complete problems, which are the hardest problems in NP.

Definition 2.2. We say that a problem $\mathcal{L}$ is NP-*hard* if every problem in NP is (polynomial-time many-one) reducible to $\mathcal{L}$. We say that $\mathcal{L}$ is NP-*complete* if $\mathcal{L}$ is NP-hard and also belongs to NP.

A standard way of showing that a problem $\mathcal{L}$ is NP-hard is to devise a reduction from another NP-hard problem. Then, to prove NP-completeness, it is enough to show that $\mathcal{L}$ is in NP.

2.5 Parameterized Computational Complexity

The central goal of parameterized computational complexity, which is briefly described in this section, is to find a way of circumventing the conjectured "practical intractability" of NP-hard problems. The general approach introduced by Downey and Fellows [DF95] is to identify a particular feature—a *parameter*, which is usually a natural number—of a problem and then to measure the problem's computational complexity with respect to both the instance size and the parameter value. The hope is that the problem becomes "tractable" for certain values of the parameter. Here, we only give a primer on computational complexity and we omit all proofs; we refer to standard textbooks [Cyg+15, DF13, FG06, Nie06] for the details.

Definition 2.3. Let Σ be a finite alphabet. Then, a set $\mathcal{L} \subset \Sigma^* \times \mathbb{N}$ is a *parameterized problem* and, for an instance $\mathcal{I} = (x, k)$, k is the *parameter value*.

Since no polynomial-time algorithm for any NP-hard problem is known, it seems that a superpolynomial computational complexity for NP-hard problems is unavoidable. This is where the tractability concept for parameterized problems plays a central role. Intuitively, parameterized problems are tractable if the superpolynomial growth in the computational complexity is exclusively related to the parameter value.

Definition 2.4. A parameterized problem $\mathcal{L}$ is *fixed-parameter tractable* with respect to parameter k if, for every instance instance $\mathcal{I} = (x, k)$, it is solvable in running time $f(k) \cdot \langle \mathcal{I} \rangle^c$ for some computable function f, some constant c, and $\langle \mathcal{I} \rangle$ being the length of encoding of $\mathcal{I}$. The class of fixed-parameter tractable problems is called FPT.

For reasons of brevity, throughout the thesis we sometimes say that an algorithm runs in FPT-*time* with respect to some parameter k if it solves every instance of some fixed problem in a running time compliant with the criteria of fixed-parameter tractability from Definition 2.4. To show a canonical example of a problem solvable in FPT-time, let us first define the Vertex Cover problem.

> Vertex Cover
> **Input:** An undirected (simple) graph $G = (V, E)$ with a set V of vertices and a set E of edges, and a positive integer k.
> **Question:** Is there a vertex cover $C \subseteq V$ of size k, that is, a subset of k vertices of G such that every edge in E is incident to at least one vertex in C?

Observe that Vertex Cover requires that for each edge at least one of its endpoints is in a sought vertex cover. Thus, we can construct a search tree in which starting from an arbitrarily chosen edge, we take one of its endpoints, remove all edges incident to the chosen endpoint, and then repeat the process until we either find a vertex cover or we select more than k vertices. Indeed, since the depth of the described search-tree is at most k and at each step we branch into at most two choices of an endpoint to analyze, we obtain an algorithm running in time $O(2^k(|E|+|V|))$. Hence, the presented algorithm runs in FPT-time with respect to the parameter "vertex cover size" (denoted by k in our definition of Vertex Cover).

The class FPT is an analogue of the class P from classical complexity theory. Similarly, the hierarchy of classes W[t], for $t \in \mathbb{N}$, plays a role similar to that of NP in classical complexity theory. Namely, it is conjectured that $\mathsf{FPT} \subset \mathsf{W}[1] \subset \mathsf{W}[2] \subset \ldots$. Again, as in classical complexity theory, a concept of reductions serves as a way to define respective W[t]-hardness notions and exclude fixed-parameter tractability.

Definition 2.5. Let Σ be a finite alphabet. A *parameterized polynomial-time many-one reduction* from a parameterized problem $\mathcal{L} \in \Sigma^* \times \mathbb{N}$ to a parameterized problem $\mathcal{L}' \in \Sigma^* \times \mathbb{N}$ is a function $f\colon \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that for every instance $\mathcal{I} = (x,k) \in \Sigma^* \times \mathbb{N}$, instance $\mathcal{I}' = (x',k') = f(\mathcal{I}) = f((x,k)) \in \Sigma^* \times \mathbb{N}$, and some computable function $g\colon \mathbb{N} \to \mathbb{N}$:

1. $f((x,k))$ can be computed in time $g(k) \cdot \langle x \rangle^c$ for some constant c;
2. $k' \leq g(k)$;
3. $\mathcal{I} \in \mathcal{L} \Leftrightarrow f(\mathcal{I}) \in \mathcal{L}'$.

We say that a problem $\mathcal{L}$ is (polynomial-time many-one) *reducible in the parameterized sense* to a problem $\mathcal{L}'$ if there is a parameterized polynomial-time many-one reduction from $\mathcal{L}$ to $\mathcal{L}'$.

With the concept of parameterized reductions, showing a parameterized reduction from a W[t] problem to some parameterized problem $\mathcal{L}$ implies that $\mathcal{L}$ is (presumably) not fixed-parameter tractable. Covering all caveats (like, for example, basic complexity assumptions, similar to $\mathsf{NP} \neq \mathsf{P}$ founding classical computational complexity theory) related to the W[t] classes is well beyond the scope of this introduction. Yet, we provide a basic W[1] problem frequently used in the literature (and sometimes in this thesis) to (presumably) exclude

fixed-parameter tractability of other parameterized problems by showing their W-hardness. Specifically, it is widespread to use the CLIQUE problem parameterized by the clique size, denoted as k in the definition below.

> CLIQUE
> **Input:** An undirected (simple) graph $G = (V, E)$ with a set V of vertices and a set E of edges, and a positive integer k.
> **Question:** Is there a clique $C \subseteq V$ of size k, that is, a subset of k vertices of G such that they are pairwise adjacent in G?

We do not know a convenient characterization in terms of a function upper-bounding the time complexity for the classes $\mathsf{W}[t]$. For this reason, it is useful to define another class that is a superset of all $\mathsf{W}[t]$ classes.

Definition 2.6. A parameterized problem $\mathcal{L}$ is in class XP if there is an algorithm that, for each instance $\mathcal{I} = (x, k)$ of $\mathcal{L}$, decides whether $\mathcal{I} \in \mathcal{L}$ in time $\langle x \rangle^{kc}$ for some constant c, where $\langle x \rangle$ is the length of an encoding of x.

Naturally, it is known that $\mathsf{FPT} \subseteq \mathsf{XP}$; the strong containment is conjectured ($\mathsf{P} = \mathsf{NP}$ implies the collapse of the whole $\mathsf{W}[t]$ hierarchy).

Even stronger intractability of a parameterized problem arises if the problem remains NP-hard even if the value of the parameter is constant. We sometimes call such problems *para*-NP-*hard*.

2.6 Integer Linear Programming

A mathematical problem of finding integral (optimal) solutions to a set of linear inequalities has become one of the fundamental problems in computer science. Formally, its decision variant can be expressed as follows.

> INTEGER LINEAR PROGRAM FEASIBILITY (ILPF)
> **Input:** An integral constraint matrix A of dimension $r \times t$, a right-hand side vector $\mathbf{b} \in \mathbb{Z}^r$, and two boundary vectors $\mathbf{l} \in \mathbb{Z}^t$ and $\mathbf{u} \in \mathbb{Z}^t$
> **Question:** Is there an integral vector $\mathbf{x} \in \mathbb{Z}^t$ such that $A\mathbf{x} = \mathbf{b}$ and $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$.

In our work we also use the optimization variant of ILPF. Specifically, we define INTEGER LINEAR PROGRAMMING where the goal is to minimize a linear function with integral variables subject to a set of linear constraints.

Integer Linear Programming (ILP)
Input: An constraint matrix A of dimension $r \times t$ with all entries being integral, an right-hand side vector $\mathbf{b} \in \mathbb{Z}^r$, two boundary vectors $\mathbf{l} \in \mathbb{Z}^t$ and $\mathbf{u} \in \mathbb{Z}^t$, and a vector $\mathbf{w} \in \mathbb{Z}^t$ representing a linear goal function.
Task: Find an integral vector $\mathbf{x} \in \mathbb{Z}^t$ that minimizes $\mathbf{w}^\intercal \mathbf{x}$ subject to $A\mathbf{x} = \mathbf{b}$ and $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$.

For the sake of readability, we always provide the goal function simply as a function, not as a vector. The ILP problem is NP-hard, thus each problem in NP has an ILP formulation (computable in polynomial time with respect to the size of the encoding of the source problem's input). This fact, together with numerous useful results already present in the literature, makes ILP a quite universal tool for studying the computational complexity.

Even though in the above definition our goal is to minimize the linear function represented by $\mathbf{w}$, the variant with maximization of the goal function is equivalent. Additionally, the integral inequalities in the definition of ILP may be augmented with some linear equalities by simulating each equality by two "opposite" inequalities. (See, for example, a textbook by Schrijver [Sch86, Chapter 7] for a detailed discussion on the equivalence of different ILP definitions). Thus, in our applications of ILP, we sometimes maximize the goal function and usually also use equalities.

We end this section with a very prominent result about fixed-parameter tractability of ILP with respect to the number of variables. The following Proposition 2.1 was first shown by Lenstra [Len83] and then improved by Kannan [Kan87] and Frank and Tardos [FT87].

Proposition 2.1 ([FT87, Kan87, Len83])**.** *Let $\mathcal{I}$ be an instance of* ILP *with t variables (that is, the constraint matrix has t columns). There is an algorithm that finds an optimal solution $\mathbf{x} \in \mathbb{Z}^t$ to $\mathcal{I}$ in $O\left(t^{2.5t+o(t)} \cdot \langle \mathcal{I} \rangle\right)$ time (where $\langle \mathcal{I} \rangle$ is the input size).*

A typical use-case of Proposition 2.1 is to show fixed-parameter tractability of a problem for some parameter p by devising an ILP formulation of the problem such that the number of variables in this formulation is upper-bounded by some function $f(p)$ of the parameter.

2.7 Experimental Environment

In our thesis, we present results of several experiments we carried out to closer investigate our problems. While in Chapter 7 we simply analyze some

properties of the experimental outcomes neglecting the efficiency of our algorithms, in Chapters 5 and 8 we mainly focused on the running times; thus, in this section we briefly the (computing) environment used for the experiments from these two sections.

The experiments in Chapters 5 and 8 were conducted on machines with Intel® Xeon® W-2125 4.00 GHz processors with four cores, equipped with 256 GB of RAM memory. The machines were operated by Ubuntu 18.04.4 LTS. The experiments in Chapter 7 were partially run on different machines but, since for them we did not focus on running time, we do not provide the exact specification.

Part I

Resource Allocation. Dealing with Private Bundles

Since 1948, when Steinhaus [Ste48] was the first to ask how to fairly distribute a divisible resource (nowadays so-called "cake") among a set of agents with (possibly different) heterogeneous valuations of the resource, studying this kind of problems has gained significant attention in mathematics, economics, computer science, and the like.

Throughout the years, in the theory of fair division two basic research directions have evolved depending on the nature of the resources; namely, they can be either divisible or indivisible.[1] The former type yields the so-called cake cutting problem (we refer to the books [BT96, Mou03, RW98] and recent surveys [BCM16, Mar17, Pro13, Pro16] for details). In this part, however, we solely focus on the latter case of indivisible resources where, as intuition suggests, a resource either cannot be divided or becomes valueless if split; for instance, a piano or a pet.

Nowadays, fair allocation of indivisible goods is a fundamental topic lying in the intersection of economics and computation [BCM16, BKV18]. Unfortunately, from a computational perspective, even for special cases, it typically is a notoriously hard (NP-hard and beyond) problem [BBN16, BL08, Kei+09], motivating the efforts to overcome this hardness in various ways. In this part, we focus on approximate fairness concepts and on deriving fixed-parameter tractability results for parameters assumed to be small in relevant practical

[1] Usually all considered resources have the same nature, however there are a few works considering variants of the problem with mixed natures of the resources [AW19, Azi+19].

application scenarios. The goal of both chapters in this part is to identify special cases where we can efficiently compute exact solutions of fair allocation problems. Whereas in Chapter 5 we mostly focus on providing fixed-parameter tractability results based on integer linear programming methods, in Chapter 4, we mainly define an "approximate notion of fairness" and detect fair allocation problem variants for which it is computationally feasible to find approximate solutions.

An important aspect of problems we consider in this part is that each resource is allocated to a single agent for its exclusive use (unlike in Part II where, in principle, a single resource can be used by many agents). Such an assumption is justified for many real-life cases such as workstations, single-person dorm rooms, stock shares, and (radio) frequency bands.

CHAPTER 3

Formalism of Resource Allocations

We devote this chapter to establishing notation and formally defining what is an allocation. We also cover fundamental concepts of fairness and efficiency that are then used for presenting our results. Throughout the whole thesis we always denote resources by $\mathcal{R}$ and agents by $\mathcal{A}$.

Definition 3.1. An *allocation* of resources $\mathcal{R}$ to agents $\mathcal{A}$ is a mapping $\pi\colon \mathcal{A} \to 2^{\mathcal{R}}$ such that $\pi(a)$ and $\pi(a')$ are disjoint whenever $a \neq a'$. For every agent $a \in \mathcal{A}$, we call $\pi(a)$ the *bundle* of a under π.

Measuring fairness requires a possibility to compare how much agents like different bundles. There are different ways to model preferences of agents over resources. A well-established (and quite flexible) way, which we also focus on, is to express the preferences numerically using so-called utility functions.

Definition 3.2. For a set of resources $\mathcal{R}$, a function $u\colon 2^{\mathcal{R}} \to \mathbb{Z}$ is called a *utility function* and, for some bundle $X \in \mathcal{R}$ its output is called the *utility* of X.

In our work, we solely focus on utility functions that are additive. Additivity intuitively means that each resource has always a fixed utility, independently of other resources in a bundle.

Definition 3.3. A utility function $u\colon 2^{\mathcal{R}} \to \mathbb{Z}$ is *additive* when, for each bundle $X \in \mathcal{R}$, $u(X) = \sum_{r \in X} u(\{r\})$. We slightly abuse the notation and, for a singleton $\{r\} \subseteq \mathcal{R}$, we write $u(r)$ instead of $u(\{r\})$.

In fair allocation problems we consider, it is always the case that each agent comes with its own, private utility function over the set of resources. Thanks to additivity, each agent can report utility values solely for single items. Thus, we always simplify additive utility functions from Definitions 3.2 and 3.3 to functions that are mapping each resource to its utility, and, without introducing ambiguity, call the latter also utility functions. Note that Definition 3.3 allows for negative utility values as well as positive utility values. As we will show later in Chapters 4 and 5, negative utility values make allocation problems more complicated. Indeed, one cannot assume that enriching a bundle with one

resource increases the utility of the bundle. Indeed, this added resource might have a negative utility, which will result on decreasing the utility of the bundle.

In our work we consider allocations that are fair. There are several well-known fairness concepts studied in the literature (some of them we introduce in Chapter 4 where we need them) but we mostly focus on envy-freeness. This is reached when, after allocating the resources, there is no agent that is willing to swap its own bundle with any bundle of resources assigned to another agent.

Definition 3.4. An allocation π is *envy-free* if there are no two agents a_1 and a_2 with utility functions u_1 and u_2 such that:

$$u_1(\pi(a_1)) < u_1(\pi(a_2)). \tag{3.1}$$

CHAPTER 4

Graph Envy-Freeness

In this chapter, we focus on finding an *envy-free* allocation of indivisible resources to agents, a central task in many multiagent systems. Often, non-trivial envy-free allocations do not exist, and, when existing, finding them can be computationally hard. Classical envy-freeness requires that every agent likes the resources allocated to it at least as much as the resources allocated to *any* other agent. In many situations, this assumption can be relaxed since agents often do not even know each other. We enrich the envy-freeness concept by taking into account (directed) social networks of the agents. Thus, we require that every agent likes its own allocation at least as much as those of all its (out)neighbors. This leads to a "more local" concept of envy-freeness. We also consider a strong variant where every agent must like its own allocation more than those of all its (out)neighbors.

We analyze the classical and the parameterized complexity of finding allocations that are envy-free with respect to one of the variants of our new concept, and that are complete. To this end, we study different restrictions of the agents' preferences and of the social network structure. We identify cases that become easier (from NP-hard to P) and cases that become harder (from P to NP-hard) when comparing classical envy-freeness with our graph envy-freeness. Furthermore, we spot cases where graph envy-freeness is easier to decide than strong graph envy-freeness, and vice versa.

4.1 Introduction

Modern management strategies emphasize the role of teams and team-work. To have an effective team one has to motivate the team members in a proper way. One method of motivating team members is to reward them for achieving a milestone. On the one hand, it is crucial that every member of a team feels rewarded fairly. On the other hand, in every team there are hierarchical or personal relations, which one should take into account in the rewarding process. Since, according to the recent labor statistics in the US [Bur], the average cost of employee benefits (excluding legally required ones) is around 25% of the whole cost of labor, it is important to effectively use rewarding instruments. It

is tempting to follow a simplistic belief that tangible incentives motivate best, and thus reward employees with cash bonuses and pay raises. However, it has been shown that to keep the employee satisfaction high, an employer should also honor the employees with non-financial rewards [Hai+15].

We propose a model for the fair distribution of indivisible goods which can be used to find an allocation of non-financial rewards[1] such that each team member is satisfied with her or his rewards and, at the same time, is not worse off compared to any other peer whom he or she is in relation with. Besides the rewarding scenario, our model has numerous further potential applications; just to mention a few: targeting marketing strategies (giving non-monetary bonuses to loyal customers), allocating physical resources to virtual resources in virtualization technologies (both network and machine virtualization), and sharing charitable donations between cities or communities which may envy each other.

Returning to our initial example of reward management, it is a well-established fact that team members evaluate the fairness of rewarding based on comparisons with their peers. This phenomenon, first described seventy years ago by the social psychologist Festinger [Fes54], is probably one of the reasons of the popularity of fair allocation (division) problems in computer science. Naturally, when evaluating the subjective fairness of rewards, every team member tends to compare itself to similar peers, neglecting those who differ substantially in position, abilities, or other aspects. This has already been reflected by one of Festinger's hypotheses; however, so far, most research in computer science has focused on fairness notions based on "global" comparisons, that is, pairwise comparisons between all members of society.

In this chapter we aim at incorporating "local" comparisons into the fair allocation scenario. Having a collection of indivisible resources we look for a way to distribute them fairly among a group of agents which, prior to the distribution, shared their opinions on how they appreciate the resources. For example, imagine that a company is to reward a team of three employees responsible for a successful project. The team consists of a key account manager (KAM) being the chief of the group, an internet sales manager (ISM), and a business-to-business (B2BSM) sales manager. The company intends some non-financial rewards to recognize the employees' performances. The rewards are "participating in a language course," "being the company's representative

[1] Financial rewards can be interpreted as divisible resources while we focus on indivisible resources.

	KAM	B2BSM	ISM
language course	☐	☑	☑
TV episode	☐	☑	☑
high-end office	☑	☑	☑
employee-of-the-month award	☑	☑	☑

Table 4.1.: The results of a survey concerning the employees' preferences over the possible rewards. Checked boxes indicate the approved rewards of a particular person.

for an episode of a documentary program," "moving to a new high-end office," and "receiving an employee-of-the-month award." The employees (agents) were surveyed for their favorite rewards, yielding the results given in Table 4.1.

Each agent considers a rewarding unfair if after exchanging all its rewards with all rewards of some peer, the agent would get more approved rewards. According to the company's rewarding policy, all rewards must be handed out. Considering the standard model of resource allocation, where each agent can compare itself to each other agent, the company cannot find a fair reward allocation. At least one agent has to get two rewards. As a consequence, two employees that have one reward are envious. However, a rewarding policy in the company assumes that a team's chief is always a basis of team success and thus deserves a better reward. Hence, both sales managers do not compare their rewards to the ones of their boss. Naturally, the key account manager's reward should be at least as good as the ones of the others. To illustrate these relations, we use the directed graph depicted in Figure 4.1.

In this case, the company can reward the key account manager with the office and the employee-of-the-month award, and distribute the two remaining rewards equally to the internet and business-to-business managers. Doing so, the company achieves a fair rewarding. The key account manager has two favorite rewards and there is no incentive to exchange them. The remaining team members do not compare themselves to their boss, so they do not envy her or him. Finally, both the business-to-business and internet managers have one favorite reward, so there is no envy. Thus, by introducing the graph of relations between the employees, we are able to represent social comparisons.

Related Work Abebe, Kleinberg, and Parkes [AKP17] and Bei, Qiao, and Zhang [BQZ17] introduced social networks of agents into the fair division context.

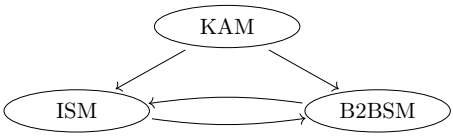


Figure 4.1.: An illustration of who compares to whom for the introductory example. Every node represents an employee and arcs represent directions of comparisons, for instance, if an arc points from the key account manager to the internet sales manager, then the former compares herself to the latter.

They defined fairness concepts based on social networks and then studied them from a computational complexity perspective. Although they defined local envy-freeness, their concept significantly differs from ours by considering divisible resources.

Strongly related to our model is a parallel work of Aziz et al. [Azi+18]. They analyzed relations of different notions of envy-freeness in the context of partial knowledge of agents (extending similar work of Bouveret and Lemaître [BL16] classifying fairness concepts in the case of full knowledge) introducing *epistemic envy-freeness*. More importantly in the context of this work, Aziz et al. [Azi+18] presented a general framework for fairness concepts which also captures our model. However, since their main goal was to introduce the framework, they did not study this specific model.

Recently, Beynier et al. [Bey+19] published a study on finding local envy-free allocations in the so-called *housing markets*, where an allocation assigns at most one resource per agent. This restriction, together with their assumptions that a given social network is undirected and that preferences are ordinal, makes their model substantially different to ours. Notably, they also studied the impact of different graph classes on the computational complexity of the problem, showing that even for very simple graphs the problem is NP-hard.

Very recently, Eduard et al. [Edu+20] took up our model and conducted a thorough study of the influence of "tree-likeness" and the density of the social network of the agents on the computational complexity of finding graph-envy-free allocations. Instead of considering the number of resources and the number of agents, they focused on the number of *resource types* and the number of *agent types*. (Two resources are of the same type if they are valued exactly the same; two agents are of the same type if they value all resources exactly the same.) It turned out that even for social networks very similar to trees, finding

graph-envy-free allocations is W[1]-hard with respect to the combined parameter "number of resource types, number of agent types, and maximum bundle size."

Distributed allocation of indivisible resources is another variant of the allocation problem that has been recently studied in the context of social relations between agents. In this distributed version, an allocation, instead of being executed by a central mechanism, emerges from a sequence of trades between the agents initially endowed with some resources. Gourvès, Lesca, and Wilczynski [GLW17] considered housing markets with a social network describing the possible agent interactions. They addressed the computational hardness of several questions such as existence of a Pareto-efficient allocation, reachability of a particular allocation, or reachability of a resource for a candidate. They proved that answering these questions is NP-hard in general, but it is polynomial-time solvable for some restricted cases. Their model has been further studied by Bentert et al. [Ben+19] and Saffidine and Wilczynski [SW18]. Chevaleyre, Endriss, and Maudet [CEM17] enriched the distributed allocation problem with monetary payments for the trades and the initially possessed resources. They defined a version of graph envy-freeness which takes into account not only allocations of resources as in our case but also the payments of agents. They showed several results describing convergence of trades leading to a fair allocation and that finding a sequence of trades reducing unfairness among the agents is NP-hard.

A somewhat orthogonal model where relations of resources, instead of agents, are described by a graph has also been studied recently [Bei+21, Bil+19, Bou+17, IP19, Suk17]. The main focus of this line of work is to study allocations that assign to agents only bundles that form connected components with respect to the given graph.

Our model is also related to a work of Gourvès, Monnot, and Tlilane [GMT18] who introduced and studied the computational complexity of the Subset Sum with Digraph Constraints problem. Their main motivations were applications in job scheduling and the issue of updating modular software. By adding solution constraints encoded as a directed acyclic graph, they generalized standard Subset Sum obtaining a variant similar to our model for the case of identical preferences.

In contrast to previously studied models, in our work, we assume that allocations are computed by a central authority and that each agent can obtain more than one resource. Furthermore, the resources we study are not subject to monetary payments to agents as a compensation for not obtaining a resource.

Our Contributions Our work follows the recent trend of combining fair allocation with social networks. We introduce social relations into the area of fair allocation of *indivisible* resources *without monetary payments*. Making use of a greater model flexibility resulting from embedding agents into a social network, we define two new versions of the classical envy-freeness property; namely, (weak) graph envy-freeness and strong graph envy-freeness. Even though Chevaleyre, Endriss, and Maudet [CEM17] also introduced a property called graph envy-freeness, their version differs from ours. Namely, instead of being a property of an allocation, it describes a particular state of the negotiations between the agents and includes monetary payments paid to the agents or by the agents in the negotiations so far. These differences (mainly monetary payments) result in the fact that in their framework, under mild assumptions, efficient and strong graph envy-freeness always exist. This stands in sharp contrast to our setting of indivisible resources, where efficient envy-free allocations might not exist. Moreover, as we will see, it is usually NP-hard to determine this.

We study problems concerned with finding (weakly/strongly) graph-envy-free efficient allocations employing completeness as the efficiency criterion. We assume that the agents' preferences over the resources are cardinal, additive, and monotonic. We go beyond the general case (with no further constraints on agent preferences and an arbitrary social network), and we analyze our problems with respect to social networks being directed acyclic graphs or strongly connected, and with respect to identical or 0/1 preferences over the resources. As a result, we explore a broad and diverse landscape of the classical computational complexity of the introduced problems. Our results reveal that in comparison to classical envy-freeness, our model sometimes simplifies the task of finding a proper allocation and sometimes makes it harder. Similarly, we identify cases where finding a (weakly) graph-envy-free allocation is easier than finding a strongly graph-envy-free allocation but also cases where the opposite is true. Additionally, our work assesses the parameterized computational complexity of several cases with respect to a few natural parameters such as the number of agents, the number of resources, and the maximum number of neighbors of an agent.

Organization In the following sections, after covering necessary preliminaries (Section 4.2), we formally introduce our new model, discuss it, and present the corresponding computational problems (Section 4.3). Then, we analyze the problem of finding (weakly) graph-envy-free allocations that are complete

(Section 4.4) followed by a study on seeking strongly graph-envy-free allocations (Section 4.5). We end with conclusions and suggestions for future work (Section 4.6).

4.2 Basic Definitions

In this section, we provide several graph-theoretical definitions regarding directed graphs followed by definitions of the fairness and efficiency concepts we use in this chapter.

4.2.1 Paths, Connected Components, and Condensations

In all definitions below, we use some arbitrary directed graph $G = (V, A)$ where V is a set of vertices and A is a set of arcs.

Definition 4.1. A *directed path* in a graph G is a sequence $S = (e_1, e_2, \ldots, e_k)$ of distinct arcs of G such that there is a sequence $v_1, v_2, \ldots, v_{k+1}$ of vertices of G such that for each $i \in [k]$, it holds that $(v_i, v_{i+1}) \in S$. If such a path exists in G, then we say that G *contains* a (directed) path of *length* k from v_1 to v_{k+1}.

Interchangeably with "G contains a path," we say that "there is a path in G."

Definition 4.2. A graph $G = (V, A)$ is *strongly connected* if, for each pair of distinct vertices $v, v' \in V$, G contains a directed path from v to v'.

Sometimes, it is convenient to work with directed graphs as collections of strongly connected parts.

Definition 4.3. Let $G = (V, A)$ be a directed graph. A graph $G' = (V', A')$ with $V' \subseteq V$ and $A' \subseteq A$ is a *subgraph* of G.

Definition 4.4. Let G be a directed graph and G' with a vertex set V' be a strongly connected subgraph of G. Graph G' is a *strongly connected component* of G if it is inclusion-wise maximal, that is, there is no vertex $v \in V \setminus V'$ from which, simultaneously, there is a path in G to the vertices in G' and there is a path in G from each vertex in G' to v.

Eventually, a partition of a graph into its strongly connected components together with arcs reflecting connections between them yields the so-called condensation of the graph.

Definition 4.5. Let G be a directed graph and G' with a vertex set V' be a strongly connected component of G. We *contract* G' by deleting all vertices

from V' (and their incident arcs) from G and adding a new vertex v^* (and the corresponding incident arcs) to G such that $N^+(v^*) = \bigcup_{v \in V'} N^+(v)$ and $N^-(v^*) = \bigcup_{v \in V'} N^-(v)$.

Definition 4.6. A *condensation* of a directed graph G is a directed graph obtained by contracting every strongly connected component in G.

4.2.2 Preferences, Fairness, and Efficiency

We continue with defining standard concepts needed to formally introduce our problems. In all subsequent definitions, we refer to a set of resources as $\mathcal{R}$ and to a set of agents as $\mathcal{A}$. In this chapter we focus on numerical utility functions that, in addition to being additive (see Definition 3.3), are also monotonic (and sometimes we restrict them even further).

Definition 4.7. An additive utility function $u\colon \mathcal{R} \to \mathbb{Z}$ is *monotonic* if it maps to non-negative utilities only. A utility function is a 0/1 *utility function* if, for each $r \in \mathcal{R}$, $u(r)$ is either zero or one.

For convenience, throughout this chapter, we frequently refer to additive monotonic or 0/1 preferences instead of saying, for instance, "preferences expressed by additive monotonic utility functions." In the problems we study (introduced in Section 4.3), we always speak about multiple utility functions representing agent preferences (one function per agent) and we, call preferences *identical* if every corresponding agent has the same utility function.

Having defined preferences, we formally present our graph fairness concepts based on comparisons between neighbors in a social network.

Definition 4.8. Let $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ be a directed graph, called an *attention graph*, representing a social network over the agents (i.e., the agents are the vertices of $\mathcal{G}$). We call allocation π *(weakly) graph-envy-free* if for each pair of (distinct) agents $a_1, a_2 \in \mathcal{A}$ such that $a_2 \in N^+(a_1)$ it holds that $u_1(\pi(a_1)) \geq u_1(\pi(a_2))$. By replacing the weak inequality in our criterion with a strict inequality, we obtain the definition of a *strongly graph-envy-free* allocation.

Naturally, an allocation which gives nothing to every agent is always (weakly) graph-envy-free. To overcome this trivial case, we combine our fairness concepts with the concept of completeness, which, intuitively, forbids leaving any resources unassigned.

Definition 4.9. An allocation π of a set $\mathcal{R}$ of resources to a set $\mathcal{A}$ of agents is *complete* if $\bigcup_{a \in \mathcal{A}} \pi(a) = \mathcal{R}$.

The notion of completeness has very natural real-life interpretations. First, complete allocations guarantee that the resources are not "wasted" by not assigning them. Second, if one assumes that disposing unallocated resources comes at a price, then completeness guarantees that the price of disposing unallocated resources is zero. Due to the second interpretation, complete allocations are sometimes referred to as allocations *without free disposal* (see Bei, Huzhang, and Suksompong [BHS20] for an example in the fair division domain).

4.3 Model and Discussion

The section is devoted to the model we introduce, a discussion on the model and its variants, and a collection of basic observations that give an initial intuition about problems that we consider.

4.3.1 Computational Problem

The core of our investigations is a computational problem related to our setting of fair allocation. We define our problems in the form of search problems instead of decision problems. In practical applications in which fair allocation problems are usually found, it is important not only to know that there exists an allocation with particular features, but also to know how it looks like. Clearly, all our problems also have natural decision variants.

C-GEF-Allocation (resp. C-sGEF-Allocation)
Input: A set $\mathcal{A}$ of n agents, a set $\mathcal{R}$ of m indivisible resources, a family $U = \{u_1, u_2, \ldots, u_n\}$ of agent non-negative utility functions, and a directed graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$.
Task: Find a complete and graph-envy-free (resp. strongly graph-envy-free) allocation of $\mathcal{R}$ to $\mathcal{A}$.

Our definitions of C-GEF-Allocation and C-sGEF-Allocation already assume additive, monotonic utility functions since this is the type of preferences we focus on. We remark, however, that this assumption could be relaxed to achieve more general problems. Yet, many of our positive (polynomial-time solvability and fixed-parameter tractability) results would not apply any more. Throughout the chapter, in the running text, we abbreviate the problem names to C-GEF-A and C-sGEF-A, respectively.

4.3.2 Discussion

As we already mentioned while defining completeness in Section 4.2.2, for studying envy-freeness one needs to additionally require that an allocation is in some sense efficient. This disallows wasting all resources and, as a result, obtaining a trivial "empty," yet envy-free allocation. Apart from completeness, other prominent efficiency notions are Pareto-efficiency and maximization of the (utilitarian) social welfare. Intuitively, an allocation is Pareto-efficient if one cannot reallocate the resources such that one agent is happier and all others are not worse off after the reallocation (we focus on Pareto-efficiency in the next Chapter 5). An allocation maximizes the (utilitarian) social welfare if every resource is given to an agent that values this resource the most. In our setting of additive monotonic utilities (and assuming that there are no "valueless" resources for which every agents reports utility zero), completeness is the weakest of the mentioned notions in a sense that every Pareto-efficient allocation and every allocation maximizing the (utilitarian) social welfare is also complete. Consequently, there is a trade-off between the existence of an envy-free efficient allocation and the strength of the required efficiency concept. So, in practice, choosing an efficiency concept suitable for an application might be nontrivial and is a problem on its own. Since we do not particularly focus on this issue in our work, while discussing our model in this section, we take the "least restrictive" path and we require all allocations to be complete.

Our work studies the computational complexity of finding envy-free allocations from two main perspectives. The first one is the nature of preferences that agents report for different resources. The second one is the structure of agent relations in terms of their awareness or knowledge of each other.

Preference Domains We study the cardinal preferences that are additive and monotonic. This type of preferences is considered as a reasonable trade-off between expressive power and elicitation simplicity. However, in the domain of fair allocation, this type of preferences usually leads to (computationally) very challenging problems.

In this chapter, we consider three constrained types of preferences to track down how the problem's hardness is related to the constraints. To this end, we study identical preferences, 0/1 preferences, and identical 0/1 preferences. At first glance, these constraints might seem too strong to yield a practical model. However, apart from being widespread in the fair allocation literature, they are also practically motivated. Assuming, for example, that agents are humans, it

is rather tedious and error-prone for an agent to assign an arbitrarily chosen number to a resource. This, in effect, makes it harder to collect valid utilities. So, it might be desirable to just let the agents choose whether they like or dislike a particular resource making the process less painful and more reliable; such scenario is modeled by 0/1 preferences. Actually, it even can be impossible to survey all agents. Then, collecting preferences from a sample of all agents, averaging them and then using the averaged ones for all agents naturally leads to identical preferences.

In the most restricted variant of identical 0/1 preferences, one can in fact think of giving indistinguishable resources to the agents. Such a scenario is natural when there are a lot of resources of the same type. An extreme case of 0/1 preferences, would be considering a unit of money as a single resource. In this light, the case of identical 0/1 preferences is interesting because it could serve as a fallback each time a set of indivisible resources cannot be allocated fairly. Then, selling the resources and allocating the obtained money makes them somewhat "more divisible," which may allow for a fair allocation. Observe, that this still is not identical to the case of cake-cutting since, clearly, one cannot divide money indefinitely.

Relations Structure The concept of graph envy-freeness is a more general version of the standard envy-freeness concept—graph envy-freeness is equivalent to envy-freeness if the given attention graph is a complete graph. In another extreme case, if the attention graph is an edgeless graph, then graph envy-freeness is purposeless, for it does not impose any constraints. In the former case—that is, seeking (complete) envy-free allocations—we know that the corresponding problem is computationally challenging. Obviously, in the latter case the problem boils down to finding any complete allocation and becomes trivial. Hence, the major focus of this work is to nail down the computational complexity of finding complete and (weakly/strongly) graph-envy-free allocations for attention graph structures between these two extremes.

Our motivation, however, is not purely theoretical. Associating the attention graph over agents with their "social relations," "attention relations," or their "knowledge" of each others brings our studies closer to the real world. From this perspective, the graph classes under our consideration—directed acyclic graphs, strongly connected graphs, and general graphs—represent different situations that occur in reality. Directed acyclic graphs are suitable to cover different kinds of hierarchical structures, for example, a corporation's employees

or departments. Strongly connected graphs model non-scattered or coherent communities, where there are no clearly separated parts; just to mention groups of classmates, friends, or teammates. Agent relations can also form structures that are beyond the two above mentioned cases, which justifies analyzing general graphs as attention graphs. Consider, for example, a professional association divided into local branches. Here, most probably, the relations between agents form a so-called small-world network [WS98]. More precisely, there are some attention relations between prominent members of different branches, yet the low-ranked members of a local branch pay attention only to each other, featuring a specific "clustering" effect.

The above interpretation of the attention graph might seem arguable when compared to our choice of the attention graph being directed. However, we find it very likely that, especially for knowledge or attention relations, such a relation might be one-directional. As in our introductory example, it seems natural that subordinates rather do not envy their bosses (at least to a reasonable level). Also, in the case of knowledge, asymmetric information is not uncommon. Consider so-called "social media influencers" who are people highly visible in the social media and who are paid by companies to market their products. Influencers' social-media followers definitely know a lot more of the influencers' personal lives than the other way around.

We point out that our graph envy-freeness concept is designed in a way that an agent totally neglects resources that were not assigned to it and its neighbors in the attention graph (a model where such resources are not neglected is defined and briefly analyzed by Aziz et al. [Azi+18]). As a result, an interesting situation can occur if an agent gets nothing. Such an agent can still be not envious (for example, when the agent has no neighbors in the attention graph), even though it is clear that there are some resources that could have been assigned to the agent such that the agent would be better off. This phenomenon might be considered as a flaw in modeling fairness. However if a central authority that assigns resources is trusted, then even such an agent that gets nothing might feel comfortable. Moreover, the agents might also be very committed and agree that the situation happened for a greater good or they might be "emotionless" (non-human agents).

4.3.3 Basic Observations

We start with a technical observation saying that graph envy-freeness can be checked in polynomial time. It is enough to compare each agents' own bundle value to the values the agent assigns to its neighbors' bundles.

Observation 4.1. *Given a set $\mathcal{R}$ of resources, a set $\mathcal{A}$ of agents with additive monotonic utility functions over resources in $\mathcal{R}$, and some allocation $\pi\colon \mathcal{A} \to 2^{\mathcal{R}}$, one can decide in polynomial time whether π is (weakly/strongly) graph-envy-free.*

Proof. It suffices to compute the utility every agent associates with its bundle and then compare it to the utilities the agent assigns to the bundles of its neighbors. □

Due to Observation 4.1—showing containment of C-GEF-A and C-sGEF-A in NP—every NP-hardness and W[1]-hardness proof in Sections 4.4 and 4.5 (in our work every W[1]-hardness proof also yields NP-hardness) also implies NP-completeness of the corresponding decision problem discussed in the proof.

Intuitively, the resources that have no value for each agent are meaningless for the concept of (weakly/strongly) graph-envy-free allocation. In the following observation, we formally show that indeed we can rule them out in the first place.

Observation 4.2. *Consider an instance I of* C-GEF-Allocation *or* C-sGEF-Allocation. *Without loss of generality, in I, there are only resources to which at least one agent assigns positive utility.*

Proof. Assume that I with an agent set $\mathcal{A}$ and a resource set $\mathcal{R}$ contains a resource $r \in \mathcal{R}$ that has utility zero for all agents. Let π be some complete allocation of $\mathcal{R}$ to $\mathcal{A}$. Consider the (unique) allocation π' with exactly the same bundles as those of π but excluding resource r. Since, for each pair of (not necessarily distinct) agents $a, a' \in \mathcal{A}$, $u_a(\pi(a')) = u_a(\pi'(a'))$, it must follow that π is (weakly/strongly) graph-envy-free if and only if π' is (weakly/strongly) graph-envy-free. □

Observation 4.2, albeit simple, results in a useful consequence for the case of identical 0/1 preferences; namely, C-GEF-A and C-sGEF-A boil down to distributing a certain number of indistinguishable resources.

Observation 4.3. *Consider an instance of* C-GEF-Allocation *with m resources, identical utility functions, and graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ with some graph-envy-free allocation π. If agent b gets no resource in π, then every agent reachable from b also gets no resource in π.*

Proof. Let a be an agent with no resource in allocation π. We give an inductive argument with respect to the distance from a. Consider a base case of an

agent a' reachable from a with distance one (i.e., $(a, a') \in \mathcal{E}$) that gets at least one resource in π. Because the resources are identical and all of them have a positive value (see Observation 4.2), agent a must envy a' which contradicts that π is graph-envy-free. Let us now consider an agent a' at distance k from a. Let b be an agent at distance $k-1$ from a such that $(b, a') \in \mathcal{G}$. By applying the induction hypothesis, we know that agent b has no resource. Then, due to the base case of distance one, the same holds for a', which proves the hypothesis for distance k. □

4.3.4 ILP Models of the Problems

We present two ILP formulations (out of many possible ones), one for C-GEF-A and one for C-sGEF-A, that we will utilize in order to show fixed-parameter tractability in several subsequent proofs in this chapter. The two models are almost identical, so we first provide the model for C-GEF-A and then describe a small change leading to the one for C-sGEF-A.

To devise the ILP model for C-GEF-A, we fix an instance of C-GEF-A consisting of agents $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, resources $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$, a utility functions family $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$, and an attention graph $\mathcal{G} = \{\mathcal{A}, \mathcal{E}\}$. For some resource r, a *type* of r is a vector $t_r \coloneqq (u_{a_1}(r), u_{a_2}(r), \ldots, u_{a_n}(r))$. By $T \coloneqq \{t_r \colon r \in \mathcal{R}\}$, we denote the set of all possible types of the resources and, for each $t \in T$, by $\#t$ the number of resources of type t.

Our ILP model consists of the following variables. For each agent $a_i \in \mathcal{A}$ and each type $t \in T$, we use an integral non-negative variable x_i^t. The value of x_i^t represents the number of resources of type t given to agent a in a sought allocation. Using the introduced variables, we model C-GEF-A using the following ILP program (we do not include any goal function, since it is enough to find any feasible solution):

$$\forall t \in T\colon \quad \sum_{i \in [n]} x_i^t = \#t \tag{4.1}$$

$$\forall (a_i, a_j) \in \mathcal{E}\colon \quad \sum_{t \in T} x_i^t \cdot t[i] \geq \sum_{t \in T} x_j^t \cdot t[i] \tag{4.2}$$

Inequalities (4.1) ensure that a sought allocation is complete, while Inequalities (4.2) guarantee weak graph envy-freeness.

We obtain the model for C-sGEF-A, by adding 1 to the right-hand side of the weak inequality in (4.2).

4.4 Finding Weakly Graph-Envy-Free Allocations

We analyze the classical complexity and the parameterized complexity (Table 4.3) for finding allocations that are complete and (weakly) graph-envy-free. We identify cases where using our graph-based envy-freeness concept leads to decreased complexity (from NP-hard to P) and cases where it leads to increased complexity (from P to NP-hard), each time comparing to classical envy-freeness.

As a warm-up, we consider the case where the attention graph is acyclic. As mentioned in Section 4.3, such an attention graph can describe hierarchical dependencies between agents well. In Observation 4.4, we show that for this scenario C-GEF-A can be solved in linear time. Although the solution is straightforward (allocating all resources to agents without incoming arcs in the attention graph), Observation 4.4 provides a good starting point for further studies on detecting more polynomial-time solvable cases.

Observation 4.4. C-GEF-Allocation *for monotonic additive preferences and an acyclic input graph is solvable in linear time.*

Proof. For an acyclic directed graph $\mathcal{G}$, there is always a complete and graph-envy-free allocation that allocates all resources to some arbitrary source agent a^*: In such allocation, no agent can envy some out-neighbor because out-neighbors always get the empty bundle. A source agent (without incoming arcs) can be found in linear time. □

As the next step, we show that restricting the preferences to identical 0/1 preferences also makes the corresponding variant of C-GEF-A polynomial-time solvable for an attention graph that is strongly connected. Here, because of transitivity of the "greater or equal" relation, we obtain a simple tractable case where all agents must obtain the same number of resources. To show this, we start with the following Observation 4.5.

Observation 4.5. *Let $\pi\colon \mathcal{A} \to 2^{\mathcal{R}}$ be a graph-envy-free allocation for the case of identical utility functions. Then, for every pair $\{a, a'\}$ of agents that belong to the same strongly connected component and a (universal) utility function u, it holds that (1) $u(\pi(a)) = u(\pi(a'))$, and (2) $|\pi(a)| = |\pi(a')|$ for 0/1 utility function.*

Proof. Consider an input graph which is a cycle over two agents, a_1 and a_2. For any graph-envy-free allocation π it must be true that $u(\pi(a_1)) \geq u(\pi(a_2))$ and $u(\pi(a_2)) \geq u(\pi(a_1))$. Thus, $u(\pi(a_1)) = u(\pi(a_2))$. By an inductive argument, the last equation holds for every pair of agents in a cycle of any length. Moreover,

adding new edges connecting agents being part of a cycle does not change the situation because the relation "greater than or equal to" is transitive and reflexive. Combining this result with Observation 4.2, we conclude that we need to give every agent the same number of resources. □

It is not hard to see that the proof of Observation 4.5 in fact yields a simple algorithm solving the variant of C-GEF-A in question.

Corollary 4.6. C-GEF-Allocation *for identical* 0/1 *preferences and an input graph being strongly connected is solvable in linear time.*

Proof. Using Observation 4.5, our algorithm checks whether the number of resources is divisible by the number of agents and returns true if and only if this is the case. □

Contrasting the case of an acyclic attention graph (see Observation 4.4), restricting preferences to identical 0/1 preferences does not guarantee that the corresponding variant of C-GEF-A becomes polynomial-time solvable in general. We obtain Theorem 4.7, showing that even with identical 0/1 preferences, GEF-Allocation becomes intractable as soon as an attention graph is not strongly connected, by utilizing the second point of Observation 4.5. This point allows us to view agents from the same strongly connected component as a "uniform block of agents," which then constitutes an important building block of the proof of Theorem 4.7.

Theorem 4.7. C-GEF-Allocation *for identical* 0/1 *preferences is* NP*-hard even if each vertex has out-degree at most two and it is* W[1]*-hard for the parameter "number of resources."*

Proof. We prove Theorem 4.7 by giving two very similar many-one polynomial-time reductions from Clique. We first show the general scheme of the reduction and prove its correctness. Then, tailoring the scheme to particular cases, we indeed show the NP-hardness when each vertex has out-degree at most two and the W[1]-hardness for the parameter "number of resources."

Construction Consider a Clique instance formed by an undirected graph $G = (V, E)$ with a set $V = \{v_1, v_2, \ldots, v_n\}$ of vertices and a set $E = \{e_1, e_2, \ldots, e_m\}$ of edges, where we seek a clique of size k, that is, a set of k pairwise adjacent vertices. Without loss of generality, assume that $1 < k < n$ and $m > \binom{k}{2}$.

We present a polynomial-time many-one reduction from Clique to C-GEF-A using a special variable $x \in \mathbb{N}$, $x \geq k^2$, that will be defined later in order to show both statements of the theorem. We introduce $x^4 + nx + m$ agents and $x^4 + kx + \binom{k}{2}$ resources which are assigned utility one by each agent. We specify an input graph $\mathcal{G}$ over the agents in two steps. First, we define strongly connected components of $\mathcal{G}$ and then we add arcs connecting them. By connecting two strongly connected components we mean adding an arc between two arbitrarily chosen vertices, one from each connected component. In a first step, we build the following strongly connected components:

1. We introduce a *root component* $\mathcal{G}^*$ which consists of x^4 vertices;
2. for each vertex $v \in V$, we introduce a *vertex component* $\mathcal{G}_v$ which consists of x vertices;
3. for each edge $e \in E$, we introduce an *edge component* $\mathcal{G}_e$ which consists of one vertex.

Then, we connect the strongly connected components to form $\mathcal{G}$ of the C-GEF-A instance. Figure 4.2 depicts graph $\mathcal{G}$ resulting from the following steps:

1. For each edge $e = \{v', v''\} \in E$, we connect $\mathcal{G}_{v'}$ and $\mathcal{G}_{v''}$ to edge component $\mathcal{G}_e$ (with an arc pointing to $\mathcal{G}_e$);
2. We connect the root component with every vertex component (with an arc starting at the root component).

For the sake of readability we extend the concept of envy from agents to sets of agents. We say that a strongly connected component A' envies another strongly connected component A'' if there exists an agent from A' that envies an agent from A''. For identical 0/1 preferences, a solution to C-GEF-A has to allocate exactly the same number of resources to every agent being a part of the same strongly connected component (Observation 4.5 (2)). Thus, we say that we are allocating some number of resources to a strongly connected component (instead of an agent) when we uniformly distribute these resources to the agents that belong to the component.

Correctness We claim that there is a k-clique in G if and only if there is a complete and graph-envy-free allocation for the constructed C-GEF-A instance. Assume that there is a k-clique $C = (V_C, E_C)$ in graph G. We create a complete and graph-envy-free allocation as follows:

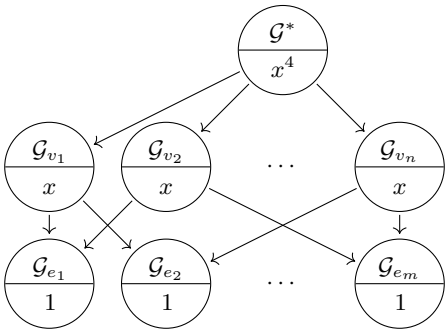


Figure 4.2.: An illustration of the general construction of $\mathcal{G}$ in the proof of Theorem 4.7. The circles represent strongly connected components. Labels indicate the name (upper part) and the number of agents in the component (lower part). The connections represent arcs between two arbitrarily chosen agents from different components.

- We give x^4 resources to agents in $\mathcal{G}^*$, a resource per agent;
- we give x resources to every agent in every vertex component associated with a vertex from V_C, a resource per agent;
- we give one resource to every agent in every edge component associated with an edge from E_C, a resource per agent.

The allocation is complete because we assign

$$x^4 + |V_C|x + |E_C| = x^4 + kx + \binom{k}{2}$$

resources. No agent in an edge component has an outgoing arc; hence, by definition, no edge component envies. Every vertex component $\mathcal{G}_v$, $v \in G$, may envy only edge components it is connected to. If $v \in V_C$, then no vertex in $\mathcal{G}_v$ envies anybody, because every vertex in $\mathcal{G}_v$ has one resource and every vertex of every edge component has at most one resource. If $v \notin V_C$, then v cannot envy because all edge components representing v's incident edges, which are not a part of clique C, have no resource allocated. Finally, the root component does not envy because each of its agents gets one resource and no other agent gets more.

Conversely, assume that there exists a complete and graph-envy-free allocation for the constructed instance of C-GEF-A. Observe that the root component can have no resources if and only if every vertex component has no resources. This in turn is impossible because if each vertex component has no resources, then every edge component also cannot have resources as this would make at least one vertex component envious. Thus, the root component has to get some resources. So, on the one hand, the root component gets at least x^4 resources because it consists of this number of agents. On the other hand, because of a lack of resources, the root component cannot get $2x^4$ resources. This derives from the following calculations using the fact that, by definition, $x \geq k^2 \geq 4$:

$$x^4 + kx + \binom{k}{2} - x^4 \leq kx + k^2 \leq 2x^3 < x^4.$$

Thus, every agent in the root component gets one resource. Since every agent in the root component might envy all other agents (even all agents in the edge components due to transitivity of the "not less than" relation), every other agent can get at most one resource. Besides the root component's resources, there are still $kx + \binom{k}{2}$ resources left. For every feasible solution there exist exactly k vertex components whose agents have a one-resource bundle. Because

$$(k+1)x = kx + x \geq kx + k^2 > kx + \binom{k}{2},$$

one cannot allocate resources to more than k vertex components. Contrarily, if one allocates x resources to $k-1$ vertex components, then there are still $x + \binom{k}{2}$ resources left. However, since each vertex component has an arc to exactly two edge components, there are at most $2(k-1) < x$ edge components that can have at most one resource each. Thus, a feasible allocation chooses exactly k vertex components and $\binom{k}{2}$ edge components. Moreover, every vertex component has to be connected to chosen edge components. This exactly corresponds to choosing k distinct vertices and $\binom{k}{2}$ distinct edges such that every edge is incident to two of the chosen vertices.

In the final step of the proof we give concrete values for x in order to show the claims.

1. We obtain NP-hardness for C-GEF-A with identical 0/1 preferences and maximum out-degree two setting $x \coloneqq nm$. Indeed, since $nm > k\binom{k}{2} = k^2 + \frac{k+1}{2}$, x meets the requirement $x \geq k^2$. In the root component, there

are $(nm)^4$ agents, which means that it is possible to connect the root component to all vertex components using a different agent from the root component. Thus, the maximum out-degree is two.

2. We obtain W[1]-hardness for C-GEF-A with respect to the number of resources by setting $x := k^2$. With such a choice the overall number of resources is depending solely on k, which implies the W[1]-hardness.

Naturally, both choices of x allow for performing the reduction in polynomial time. □

The above theorem not only shows that identical 0/1 preferences do not guarantee polynomial-time solvability. In fact, it presents two stronger negative results. C-GEF-A with identical 0/1 preferences presumably cannot be "efficiently" solved even for few resources or even if each agent is allowed to envy at most two other agents.

The remaining hope for positive results (in the form of fixed-parameter tractability) for the case of general attention graphs and identical 0/1 preferences lies in the scenario with few agents. In the following Proposition 4.8 we show that indeed this variant of C-GEF-A is fixed-parameter tractable with respect to the number of agents. In fact, we show that the fixed-parameter tractability holds also for the case of 0/1 preferences that are not identical.

Proposition 4.8. C-GEF-Allocation *with* 0/1 *preferences is fixed-parameter tractable with respect to the parameter "number of agents."*

Proof. We show that, for the case of 0/1 preferences, the number of variables in the model from Section 4.3.4 is upper-bounded by a function of the number n of agents in an instance of C-GEF-A. Observe that, for 0/1 preferences, there are at most 2^n different resources types. Thus, the model uses at most $n \cdot 2^n$ variables. Eventually, the result is a consequence of applying a result of Lenstra [Len83] (presented in Proposition 2.1) for ILP models with a bounded number of variables. □

The intractability results set by Theorem 4.7 suggest that the restriction that an attention graph must be strongly connected should be kept in further investigations on seeking polynomial-time solvability of C-GEF-A. Thus, we relax the constraints on preferences and we allow for more values than just 1 and 0 turning to the case of identical monotonic additive preferences. However, the first point of Observation 4.5 opens up a way for a (quite straight-forward)

reduction from the NP-hard and W[1]-hard EEF Existence [BL08] problem. As a result, the following Proposition 4.9 shows that C-GEF-A for identical monotonic additive preferences is intractable (in the parameterized sense) for few agents even if each agent has at most one neighbor in the attention graph.

Proposition 4.9. C-GEF-Allocation *for identical monotonic additive preferences is* NP*-hard and* W[1]*-hard when parameterized by the number of agents even if the input graph is a cycle.*

Proof. We give a polynomial-time many-one reduction from the NP-hard problem EEF Existence with monotonic additive identical preferences studied by Bouveret and Lang [BL08]. The problem is to decide whether there exists an envy-free, Pareto-efficient allocation for a given set $\mathcal{A}$ of agents, a set $\mathcal{R}$ of resources, and monotonic additive identical utility functions of the form $u\colon \mathcal{R} \to \mathbb{N}$. For monotonic additive identical preferences it is enough that an allocation is complete and envy-free to be a solution for EEF Existence (see Bliem, Bredereck, and Niedermeier [BBN16] for a more detailed discussion). To build an instance of GEF-Allocation we take the whole input from the EEF Existence instance and we add a graph being a cycle over all the agents (in an arbitrary order). To solve EEF-Allocation every agent has to get an equally valuable bundle which is also the case for the new C-GEF-A instance. The reduction is clearly computable in polynomial time. □

Observe that Proposition 4.9 does not exclude fixed-parameter tractability for few resources (if the preferences are identical). Actually, the following Theorem 4.10 shows that C-GEF-A is fixed-parameter tractable when parameterized by the number of resources.

Theorem 4.10. C-GEF-Allocation *with identical preferences is fixed-parameter tractable with respect to the number of resources for an input graph being strongly connected.*

Proof. Assume an instance of C-GEF-A with identical preferences and a strongly connected attention graph. Let n be the number of agents and m be the number of resources. According to Observation 4.2, one can withdraw from consideration all zero-valued resources. If there are more agents than there are resources, then the answer for the instance is "no." This is a direct implication of the fact that in the case of a strongly connected input graph and identical preferences each agent has to have the same utility and there is at least one non zero-valuable resource. In the opposite case, we can test all possible n^m

allocations. Because $n^m \leq m^m$ and the test for completeness and envy-freeness in a polynomial-time task, we obtain a fixed-parameter algorithm for parameter "number of resources." □

We continue our investigations on the computational complexity of C-GEF-A considering the last yet unsettled case; namely, the case of 0/1 preferences and few resources. With the classic envy-freeness notion (or $\mathcal{G}$ being complete for C-GEF-A), finding a complete and envy-free allocation can easily be seen to be fixed-parameter tractable with respect to the number of resources (using an analogous technique as used by Bliem, Bredereck, and Niedermeier [BBN16, Proposition 1]). For graph envy-freeness however, the following Theorem 4.11 shows that the problem becomes W[1]-hard even for 0/1 preferences and a strongly connected attention graph. This result provides an example where C-GEF-A, which is tractable (parameterized by the number of resources) for the case of a complete (directed) attention graph, may become intractable if one deletes some arcs from the attention graph.

Theorem 4.11. C-GEF-Allocation *with* 0/1 *preferences and an input graph being strongly connected is* NP*-hard; it is* W[1]*-hard with respect to the combined parameter "number of resources and maximum out-degree"; it is* W[1]*-hard with respect to the parameter "number of resources"; and it is* NP*-hard even if the maximum out-degree of the attention graph is three.*

Proof. To prove Theorem 4.11, we give a polynomial-time many-one reduction from Clique to C-GEF-A. To this end, we first fix notation, then describe the construction, and eventually conclude the argument with proving the construction's correctness.

Let I be a Clique instance, where given an undirected graph $G = (V, E)$ with a set $V = \{v_1, v_2, \ldots, v_n\}$ of at least two vertices, a set $E = \{e_1, e_2, \ldots, e_m\}$ of edges, and a clique size k, we ask whether there is a set of k pairwise adjacent vertices. Without loss of generality, we assume that $2 < k < n$ and $m > \binom{k}{2}$. The reduction transfers instance I to and instance I' with agent set $\mathcal{A}$, resource set $\mathcal{R}$, utilities $\mathcal{U}$, and attention graph $\mathcal{G}$.

Construction We build the set $\mathcal{A}$ of agents of all vertices and edges of G, a set $\mathcal{D}$ of *dummy agents*, and a set C of k^3 *constraint agents*. Set $\mathcal{D}$ is the union of $n+m$ groups of k^5 distinct agents each—one group $D(v)$ per each vertex $v \in V$ and one group $D(e)$ per each $e \in E$. Hence, in total, $|\mathcal{A}| = n(k^5+1)+m(k^5+1)+k^3$.

	V	E	C	$\mathcal{D}$
$\mathcal{R}_{\mathrm{v}}$	1	0	0	0
$\mathcal{R}_{\mathrm{e}}$	1	1	0	0
$\mathcal{R}_{\mathrm{c}}^*$	0	1	1	1
$\mathcal{R}_{\mathrm{c}} \setminus \mathcal{R}_{\mathrm{c}}^*$	0	0	1	1

Table 4.2.: The utilities the agents give to the resources in the construction in the proof of Theorem 4.11.

We introduce k *vertex resources*, $\binom{k}{2}$ *edge resources*, and k^3 *constraint resources*; we refer to these sets as, respectively, $\mathcal{R}_{\mathrm{v}}$, $\mathcal{R}_{\mathrm{e}}$, and $\mathcal{R}_{\mathrm{c}}$. Additionally, we set apart $\binom{k}{2}$ (arbitrary) constraint resources that we call *distinguished constraint resources* and denote them by $\mathcal{R}_{\mathrm{c}}^*$ (naturally, $\mathcal{R}_{\mathrm{c}}^* \subset \mathcal{R}_{\mathrm{c}}$). Then, we let $\mathcal{R} := \mathcal{R}_{\mathrm{v}} \cup \mathcal{R}_{\mathrm{e}} \cup \mathcal{R}_{\mathrm{c}}$, and thus we have exactly $k^4 + \binom{k}{2} + k$ resources.

We proceed with constructing the graph $\mathcal{G}$ step by step. We refer to Figure 4.3 for a better understanding of the big picture of the construction. First, for each group of dummy agents, we create a subgraph called a *separator gadget*. For each agent $x \in V \cup E$, the separator gadget $S(x)$ is a directed cycle over all agents in $D(x)$. Then, using previously defined separator gadgets, we create one part of $\mathcal{G}$ as follows:

1. for each agent $v_i \in V$, $i \in \{1, 2, \dots n-1\}$, we arbitrarily select two distinct agents x and y from $S(v_i)$ and we create two arcs: (v_i, x) and (y, v_{i+1});

2. we select two distinct agents x and y from $S(v_n)$ and add two arcs: (v_n, x) and (y, v_1).

We proceed analogously with all agents in E. In the next step of constructing the attention graph, for each edge $e = (v_i, v_j) \in E$, we add two arcs to $\mathcal{G}$: (v_i, e) and (v_j, e). We conclude the construction by adding a directed cycle $\hat{C}$ over all constraint agents, adding an arc from each $e \in E$ to a distinct, arbitrarily chosen constraint agent, and adding an arc from an arbitrary chosen constraint agent to v_1.

The final point of the construction deals with the utilities. We use 0/1 utilities as depicted in Table 4.2.

Correctness We start proving the correctness of the reduction stating a key lemma about the constraint resources.

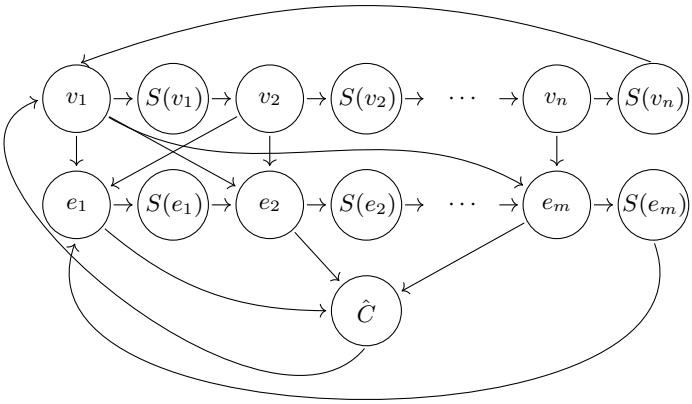


Figure 4.3.: The construction in the proof of Theorem 4.11.

Lemma 4.12. *In every graph-envy-free allocation for I', all constraint resources must be given to the constraint agents, one resource per agent.*

Proof. We give a proof by contradiction. By definition of C-GEF-A, all constraint resources must be allocated. Towards a contradiction, assume there is a graph-envy-free allocation π allocating a constraint resource to some fixed agent $a^* \notin C$. We consider the two following cases, both giving a contradiction.

1. Agent $a^* \in \mathcal{D}$. Clearly, a^* is a part of some separation gadget $S(x)$ that forms a cycle over k^5 agents. This in turn means that there is an arc $(b, a^*) \in \mathcal{G}$; thus, agent b has to also get a resource not to envy, which forces another agent, the one preceding b in the cycle, to also get another resource. This "chain reaction," in fact imposes that all k^5 agents in $S(x)$ need to get a resource. However, there are only $k^4 + \binom{k}{2} + k < k^5$ resources; hence, we get a contradiction because π cannot be graph-envy-free.

2. Agent $a^* \in V \cup E$. Then, there exists some dummy agent b such that there is an arc (b, a). Thus, for b not to envy, it has to get a resource and we arrive at the first case achieving a contradiction.

By our construction, the above cases are exhaustive and non-overlapping. □

Equipped with Lemma 4.12, we prove the correctness of our reduction. We start with showing that a clique in the original instance implies a "yes"-instance in I' and then we prove that non-existence of a clique in the original existence means that I' is a "no"-instance.

Let $\mathcal{S} = \{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_k\}$ be a clique of size k in I, and let $\hat{E} = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_{\binom{k}{2}}\}$ be the set of edges of the clique. Then, a graph-envy-free allocation π for instance I' is constructed as follows.

1. The distinguished constraint resources are given to those constraint agents whose incoming arcs come from the agents in $\hat{E}$; formally, for each $\hat{e} \in \hat{E}$, if $(\hat{e}, a) \in \mathcal{G}$, $a \in C$, then $\pi(a) = \{r\}$, $r \in \mathcal{R}_{\mathrm{c}}^*$.

2. According to Lemma 4.12, the remaining constraint resources are given to the constraint agents that have not gotten any resource yet, one resource per agent.

3. Each agent $\hat{e} \in \hat{E}$ gets a separate edge resource and each agent $\hat{v} \in \hat{V}$ gets a separate vertex resource.

Since the number of edges in clique $\mathcal{S}$ and the number of the distinguished constraint resources is the same, according to Steps 1 and 2, allocation π complies with Lemma 4.12. Observe that the edge agents can only envy agents that got a distinguished constraint resource. Thus, according to Step 1, only edge agents in $\hat{E}$ can be envious. However, in Step 3, each of these agents gets an edge resource; hence, none of them envies. By the construction, an agent $v \in V$ that has no resource envies an agent $e \in E$ such that $\pi(e) \in \mathcal{R}_{\mathrm{e}}$ if and only if $v \in e \in E$; in other words, whenever agent $e = \{v, v'\}$ gets an edge resource, agents v and v' have to get a vertex resource each. Since $\mathcal{S}$ is a clique, due to Step 1, it is indeed true that π meets this requirement. Thus, π is complete and graph-envy-free.

Next, assume that I is a "no"-instance, and for the sake of contradiction, suppose that π is a graph-envy-free allocation for I'. Due to Lemma 4.12 all constraint resources are evenly distributed among the constraint agents. Thus, what follows from the fact that there are $\binom{k}{2}$ constraint agents with a distinguished constraint resource, there is a set $\hat{E}$ of $\binom{k}{2}$ edge agents that are assigned a single edge resource by π. Let $\hat{V}$ be a set of vertices such that each vertex $\hat{v} \in \hat{V}$ has an outgoing arc pointing to at least one agent in $\hat{E}$. By the construction of $\mathcal{G}$, in fact, $\hat{V}$ are vertices in G that are incident to edges in $\hat{E}$. Thus, clearly, $|\hat{V}| \geq k$. However, if π is graph-envy-free, then $|\hat{V}| = k$ because

each agent in $\hat{V}$ has to get a vertex resource not to envy and there are k of these resources. In this case, $\hat{V}$ must be a clique, yielding a contraction.

The reduction is indeed correct and computable in polynomial-time. One can easily check that the graph $\mathcal{G}$ is strongly connected and that its maximum out-degree is at most three. Furthermore, the number of resources is a function solely of parameter k. □

4.4.1 Few Identical Resources and Small Maximum Out-Degree

We devote this section to prove Theorem 4.14. This positive result intuitively says that if there are few resources as well as agents that have identical utility functions and each agent pays attention to relatively few other agents, then C-GEF-A can be solved efficiently. We start with the following Lemma 4.13 about large connected components and then present Theorem 4.14 together with its proof.

Lemma 4.13. *Consider an instance of* C-GEF-Allocation *with m identical resources and graph $\mathcal{G}$. Assume that $\mathcal{G}$ has a strongly connected component $\mathcal{C}$ such that at least one the following holds:*

1. *the vertex corresponding to $\mathcal{C}$ has in-degree greater than m in the condensation of $\mathcal{G}$ or*
2. *component $\mathcal{C}$ has more than m agents.*

Then, an equivalent instance of C-GEF-Allocation *with an attention graph $\mathcal{G}'$ such that $\mathcal{G}'$ is a subgraph of $\mathcal{G}$ and $\mathcal{G}'$ does not contain $\mathcal{C}$ can be computed in polynomial time.*

Proof. Suppose I is an instance of C-GEF-A that meets the assumptions of Lemma 4.13. We show that a new, equivalent instance I' of C-GEF-A with a graph $\mathcal{G}'$ that does not contain the strongly connected component $\mathcal{C}$ can be constructed in polynomial time. In order to obtain I', we construct $\mathcal{G}'$ from $\mathcal{G}$ by removing component $\mathcal{C}$ and all strongly connected components reachable with a path from $\mathcal{C}$.

Due to Observation 4.5, either all agents in $\mathcal{C}$ get at least one resource or none of them gets any. In fact, in both cases of the lemma, all agents in $\mathcal{C}$ get no resource. Giving resources to all agents yields an immediate contradiction in Case 2 because of the lack of resources. In Case 1, the lack of resources is also a reason for a contradiction—due to Observation 4.3, each of the in-neighbors of $\mathcal{C}$ has to get at least one resource. So, in both cases no agent in $\mathcal{C}$ gets a resource.

Consequently, thanks to non-negative resource values, no agent that has an outgoing arc pointing to an agent in $\mathcal{C}$ can envy anymore; hence, it is safe to eliminate such arcs from $\mathcal{G}$. Furthermore, again due to Observation 4.3, every agent reachable from every agent of $\mathcal{C}$ cannot get a resource. As a result, we can safely remove $\mathcal{C}$ and all components reachable from it. Naturally, such a procedure can be applied in polynomial-time, for example, using a modification of BFS. □

Theorem 4.14. C-GEF-Allocation *with identical preferences is fixed-parameter tractable with respect to the combined parameter "number of resources and maximum out-degree."*

Proof. We give an algorithm that yields the requested fixed-tractability. A high-level idea of the algorithm is to guess a collection of bundles and the connections of agents that get the bundles. For each such a guess, the algorithm checks whether the guessed situation can be implemented in the input attention graph.

In the proof, we focus on the strongly connected components of an input graph $\mathcal{G}$, thus we merely use the condensation of an input graph $\mathcal{G}$. This suffices, since, for identical preferences, the internal structure of strongly connected components does not play any role. What is important, are only arcs between distinct strongly connected components. So, in the proof, we speak about a *bundle pack* that is a collection of bundles given to agents of the same strongly connected component.

We decompose the algorithm into the following four major steps, then we argue about their correctness and running times separately to finish the proof. The four steps read as follows (Figure 4.4 illustrates the concepts introduced by the presented steps).

1. Guess a number q of bundle packs and partition the resources into q packs $\{P_1, P_2, \ldots, P_q\}$ (thus, all packs are mutually disjoint and their union contains all resources). We call such a guess a *partial structure* of a solution.

2. To every pack P of the partial structure, assign a weight $\rho(P) \in \{1, 2, \ldots m\}$. Then, add arcs between the packs such that the arcs do not create a cycle. We obtain a vertex-weighted directed acyclic graph over the packs that fully describes a solution. Intuitively, each weight represents the number of bundles a pack consists of (and, what is equivalent, the number of agents in a strongly connected component to which the pack is given in

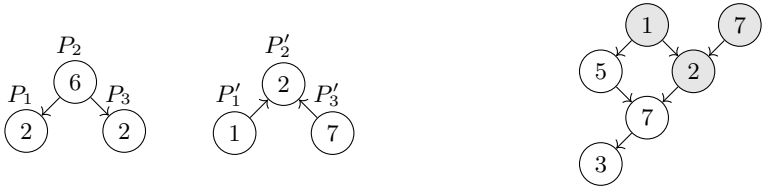


Figure 4.4.: The two example structures on the left both consist of three packs. The packs label their respective vertices in the structures. Each structure vertex (representing some pack P) is also labeled with its integer weight $\rho(P)$. For clarity, we neglect the definition of resources and utility functions. On the right, instead of the attention graph $\mathcal{G}$, we only provide its condensation. Here, each vertex label is the number of agents in the respective strongly connected component. The shaded vertices form a subgraph of the condensation that meets the criteria from Step 4 of the algorithm for the second example structure. Note that there is no subgraph of the condensation that is isomorphic to the first example structure.

the solution). The arcs represent the structure of the strongly connected components to which the packs are given. A partial structure with weights and arcs is a *structure* (see Figure 4.4 for examples).

3. Check the sanity of a guessed structure. First, for each pack P in the structure run the algorithm from Theorem 4.10 feeding it with the resources in P, a clique over $\rho(P)$ agents, and the input (identical) preferences. Then, assuming that the attention graph is the one formed by the structure and that each agent in each strongly connected component of the structure gets the same number of resources from the component's pack, check whether the structure describes a graph-envy-free allocation. If at least one of the aforementioned tests fails, then proceed with another guess. Otherwise, continue with the next step.

4. Check whether there exists a subgraph $\mathcal{G}'$ in the condensation of $\mathcal{G}$ such that $\mathcal{G}'$ has no incoming arcs from vertices outside of $\mathcal{G}'$ and it is isomorphic to the graph described by the guessed structure (including weights).

Correctness Each structure formed in Steps 1 and 2 describes an allocation where all resources are allocated. Moreover, checking all possible structures exhaustively considers all possible relations (in terms of the attention graph)

that can occur between the strongly connected components that are assigned resources.

We next show that Step 3 dismisses a guessed structure if and only if the structure does not describe a graph-envy-free allocation. Let us consider some pack P of the structure and an allocation π' that is graph-envy-free for some strongly connected component $C(P)$. We refer to the agents of the connected component $C(P)$ as $A(P)$ and we define $\rho(P) \coloneqq |A(P)|$. A direct corollary of Observation 4.16 is that (assuming identical preferences) π' is graph-envy-free for $C(P)$ if and only if it would be strongly graph-envy-free if the relations between agents in $A(P)$ were forming a complete graph. Thus, by dismissing a pack P for which there is no graph-envy-free allocation π' for a clique of agents in $A(P)$, we cannot dismiss a correct solution for the whole problem. Moreover, Observation 4.16 shows that in such an allocation π' each agent in a connected component gets exactly a bundle of the same value—an equal share. This justifies why we can safely dismiss a guessed structure if, for each pack, allocating an equal share of the resources in the pack to every agent of the pack's component does not lead to a graph-envy-free allocation considering only the relation graph described by the structure's arcs.

We reach Step 4 only if a guessed structure describes a graph-envy-free allocation assuming there are no other arcs than those in the structure. Let $\mathcal{S}$ be the (directed acyclic) graph described by the structure. Let $\mathcal{G}'$ be a subgraph of the condensation of $\mathcal{G}$ whose vertices are labeled with each strongly connected component's size. We show that the structure describes a graph-envy-free allocation for the input instance if and only if there exists such a $\mathcal{G}'$ that has no incoming arcs from vertices that are not part of $\mathcal{G}'$ and is isomorphic to $\mathcal{S}$, including weights. Let $\mathcal{G}'$ be a subgraph meeting the criteria of the claim. Because $\mathcal{G}'$ is isomorphic to $\mathcal{S}$, there is a graph-envy-free allocation π of the resources to all agents of $\mathcal{G}'$ if the arcs not present in $\mathcal{S}$ are ignored. In fact, by definition of $\mathcal{G}'$, every arc present in $\mathcal{G}'$ and not present in $\mathcal{S}$ starts in some agent in $\mathcal{G}'$ and ends in an agent not in $\mathcal{G}'$. Since every agent outside of $\mathcal{G}'$ has no items, such arcs cannot introduce envy, which means that π is also graph-envy-free for the whole input instance. Let $\mathcal{S}'$ be the directed graph of the structure describing a graph-envy-free allocation for the input instance. It is immediate that $\mathcal{G}$ contains a subgraph $\mathcal{G}'$ isomorphic (including weights) to $\mathcal{S}'$ or one of the graphs maintaining the reachability relation of $\mathcal{S}'$. Let $\mathcal{S}$ be exactly the graph that $\mathcal{G}'$ is isomorphic to (since we check all possible structures, the algorithm has to finally find $\mathcal{S}$). Indeed, if $\mathcal{G}'$ has an incoming arc from outside of its agents, then the agent from which this arc comes has to be envious; a contradiction.

Running Time Because in each solution there are at most m packs, the number of all possible partial structures is upper-bounded by m^m. Subsequently, there are at most $m^m \cdot m^4$ structures. The sanity check consists of at most m invocations of the algorithm from Theorem 4.10, which runs in FPT-time with respect to parameter m, and a single, polynomial check of the graph envy-freeness property as described in Observation 4.1. Therefore, the first three steps of the algorithm run in $f(m) \cdot \text{poly}(|I|)$ time. It remains to show that Step 4 is computable in FPT-time with respect to the number of resources plus the maximum out-degree Δ of vertices in $\mathcal{G}$.

Running Time of Step 4 The problem we need to solve in this step is very similar to Directed Subgraph Isomorphism if we take the condensation of $\mathcal{G}$ as a host graph and the guessed structure as a pattern graph. A subtle difference is that we need to ensure that a subgraph $\mathcal{G}'$ isomorphic to the guessed structure has no arcs incoming from outside of $\mathcal{G}'$. Although originally Directed Subgraph Isomorphism does not obey this constraint, we simulate it by appropriate coloring of the condensation of $\mathcal{G}$ and the guessed structure followed by solving Directed Colored Subgraph Isomorphism.

We color each vertex v, representing a pack P of the guessed structure, with in-degree $\deg^-(v)$ with a color $\lambda_s(v) := \rho(P)m + \deg^-(v)$. Similarly, we color each vertex v with in-degree $\deg^-(v)$ in the condensation graph of $\mathcal{G}$, representing a strongly connected component with ℓ agents, with a color $\lambda_c(v) := \ell m + \deg^-(v)$. Then, we solve Directed Colored Subgraph Isomorphism for such transformed graphs. The running time follows from Lemma 4.15 because each guessed structure has at most m packs, we have at most m^2 colors in the transformed graphs, and the maximum degree of the condensation of $\mathcal{G}$ is at most $(\Delta + 1)m$. □

Definition 4.10. In the Subgraph Isomorphism problem, given an undirected host graph H and an undirected pattern graph G, the question is whether there is a subgraph H' of H isomorphic to G. When graphs G and H are colored and the isomorphism has to be color preserving, then we obtain Colored Subgraph Isomorphism. Additionally, if the graphs are directed, we arrive at Directed (Colored) Subgraph Isomorphism.

Subgraph Isomorphism is a well-studied problem (see the survey of Marx and Pilipczuk [MP13] and its more detailed version [MP14] for a collection of results on the problem's parameterized complexity). Sometimes, subtasks

related to this problem are solved using the color-coding technique so Colored Subgraph Isomorphism appears "internally" as a part of proofs from time to time (see, for example, the proof of Theorem P.1 by Marx and Pilipczuk [MP13]). However, we are not aware of a publication concerning the specific relation between Directed Colored Subgraph Isomorphism and Subgraph Isomorphism that we present in the following lemma (see Lemma 2.6 by Marx and Pilipczuk [MP13] describing a very similar idea in a different context of fixing images of prescribed vertices).

Lemma 4.15. Directed Colored Subgraph Isomorphism *is fixed-parameter tractable with respect to the combined parameter "the number of vertices in the pattern graph, the number of distinct colors in the input graphs, and the maximum degree of the host graph" for directed acyclic graphs.*

Proof. A general strategy of the proof is to show a parameterized many-one reduction from Directed Colored Subgraph Isomorphism to Subgraph Isomorphism and then to use an appropriate result for the latter.

Construction Let $G^0 = (V(G^0), E(G^0), \lambda_{G^0})$ and $H^0 = (V(H^0), E(H^0), \lambda_{H^0})$ be directed colored graphs—a pattern graph and a host graph, in order, with respective vertex-coloring functions λ_G and λ_H—forming an instance I of Directed Colored Subgraph Isomorphism. We illustrate the construction's steps in Figure 4.5.

We first transfer G^0 and H^0 to undirected graphs subdividing each arc by adding two special colors c_{beg} and c_{end} marking, respectively, the beginning of an arc and the end of an arc. This transformation allows us to encode the directions of arcs. Specifically, for each arc $e = (u, v) \in E(G^0) \cup E(H^0)$, we add two vertices u' and v', and replace e with three edges $\{u, u'\}$, $\{u', v'\}$, $\{v', v\}$ setting the colors of u' and v' to c_{beg} and c_{end} respectively. In the resulting (now undirected) graph, for each edge $e = \{u, v\}$, we add a new vertex x, two edges $\{u, x\}$ and $\{x, v\}$, and delete edge e. We color the new vertex x with a new color $c_\emptyset$ that we refer to as the *void color*; we refer to all vertices colored with the void color as the *dummy* vertices and refer to the corresponding vertex-set as $V_\emptyset$. We refer to the resulting graphs and functions as G^1, H^1, λ_{G^1}, and λ_{H^1} respectively.

Second, we transfer G^1 and H^1 to non-colored graphs. Intuitively, we encode each vertex' color with the *bulb* gadget. The bulb gadget of color $i > 0$ consists of a cycle of size 3 and a cycle of size $3+i$; the two cycles have exactly one common

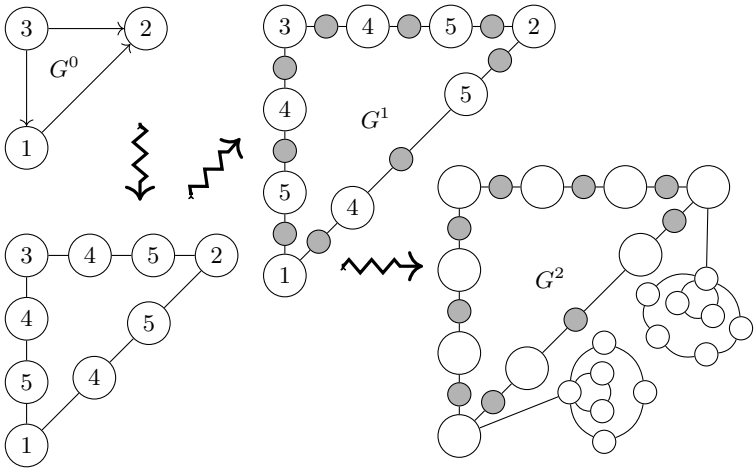


Figure 4.5.: The step-by-step construction of G^2 from G^0 described in the proof of Lemma 4.15. The unnamed graph shows an auxiliary step helping to visualize the construction. The numbers inside the vertices represent colors: $c_{\text{beg}} = 4$, $c_{\text{end}} = 5$. The dummy vertices are colored gray. For clarity, the bulbs are only demonstrated for two vertices; in fact, every non-dummy vertex has its own bulb constructed.

vertex called the *foot*. For each vertex $v \in V(G^1) \cup V(H^1)$ with color $i \neq c_\emptyset$, we construct a copy of the respective bulb gadget and connect vertex v to the bulb gadget's foot. Note that we do not create any bulb gadgets for the dummy vertices. This final transformation, together with neglecting the coloring functions, gives us undirected, non-colored graphs G^2 and H^2—these graphs form an instance I' of Subgraph Isomorphism.

Naturally, the whole construction involves polynomially-many steps, and thus the reduction runs in polynomial time.

Correctness We show that there is a directed, colored subgraph $\widehat{H^0}$ of H^0 that is isomorphic to G^0 respecting colors if and only if there is an undirected, non-colored subgraph $\widehat{H^2}$ of H^2 isomorphic to G^2.

Having $\widehat{H^0}$ we apply the same transformations as those applied to G^0 and to H^0 obtaining $\widehat{H^2}$. It is clear that $\widehat{H^2}$ is a subgraph of H^2 and that it is isomorphic to G^2.

For the reverse direction, suppose we have $\widehat{H^2}$ that is a subgraph of H^2 and is isomorphic to G^2 via an isomorphism $\eta\colon V(G^2) \to V(\widehat{H^2})$. In two steps, we will show how to transform $\widehat{H^2}$ to a directed colored subgraph $\widehat{H^0}$ of H^0 that is isomorphic to G^0. We say that a vertex is *adjacent to a bulb gadget* if the vertex is adjacent to the foot of the gadget. Similarly, we say that a vertex is *adjacent to a cycle* if the vertex is adjacent to exactly one vertex of this cycle.

First, we "bring back" colors of $\widehat{H^2}$ thus obtaining an interim graph $\widehat{H^1}$. Consider some vertex $v \in V(\widehat{H^2}) \cap V(H^1)$ that originally had color $i := \lambda_{H^1}(v)$ and some vertex $u \in V(G^2) \cap V(G^1)$ originally colored to $j := \lambda_{G^1}(u)$. We show that $\eta(u) = v$ if and only if $i = j$. To this end, we distinguish two cases depending on whether $j = c_\emptyset$. We first assume that $j \neq c_\emptyset$. By the construction, vertex u is adjacent to its respective bulb gadget in G^2. We show that if $\eta(u) = v$, then vertex v is also adjacent to a bulb of color j; from this it follows that $i = j$. First observe that no vertex $x \in V(H^2) \cap V(H^1)$ such that $\lambda_{H^1}(x) \neq c_\emptyset$ is part of a cycle of size 3. If it were the case, then there would be an edge connecting two dummy vertices or an edge connecting a dummy vertex with the foot of some bulb gadget. However, by the construction there are no such edges. Since every dummy vertex has only neighbors that are not dummy vertices (and, what we have just shown, these neighbors cannot be part of a cycle of size three), then no dummy vertex is adjacent to a vertex of a cycle of size three. As a result, v cannot be a dummy vertex and thus v is (by the construction) adjacent to a bulb. Note that v's neighbors, by definition, are only some dummy agents and the foot of v's bulb. Note that every dummy agent (by the construction) has degree two and all feet have degree exactly 5. Hence, no dummy agent can play a role of a foot. Thus, v is adjacent *only* to the foot of its own bulb. Naturally, if v is mapped to u, then it must be the case that both u and v are connected to a bulb of the same color; as a result, it holds that $i = j$. Now let us consider the remaining case where $j = c_\emptyset$, that is, u is a dummy vertex. By the construction, vertex u has two neighbors and none of them is a dummy vertex. We already know that every non-dummy vertex is mapped to another non-dummy vertex with the same color. Hence, both neighbors of u are mapped correctly. So, it follows that u has to be mapped to a dummy vertex. Thus, v is a dummy vertex and $i = j = c_\emptyset$. Eventually, since we know that the vertices are mapped correctly with respect to their colors, we get $\widehat{H^1}$ by coloring each vertex in $V(\widehat{H^2}) \cap V(H^1)$ with its respective color and removing all bulbs. Note that $\widehat{H^1}$ is a subgraph of H^1 (which is essentially H^2 with proper colors instead of bulbs) and is isomorphic to G^1 (being just G^2 with proper colors instead of bulbs).

The final step is to transform $\widehat{H^1}$ to a subgraph $\widehat{H^0}$ of H^0 isomorphic to G^0. To achieve this, we first remove every dummy vertex in $\widehat{H^1}$ by adding an edge between its neighbors (by the construction each dummy vertex has exactly two neighbors). In the second step substitute all paths of form $\{u, x, y, v\}$ in $\widehat{H^1}$ with x colored to c_{beg} and y colored to c_{end} with an arc (u, v). Such paths, by our construction, are non-overlapping and exactly encode the respective directed arcs (u, v) in H^0. Performing all substitutions we achieve $\widehat{H^0}$. Since applying the same procedure to G^1 would give G^0, it is clear that $\widehat{H^1}$ is isomorphic to G^1 if and only if $\widehat{H^0}$ is isomorphic to G^0.

Let q be the number of different colors the input graphs are colored with. Applying our reduction, we arrive at an instance I' in which the pattern graph G^2 has at most $f(E(G^0), q) := |E(G^0)|(2 + 3 + 2q + 12) + |V(G^0)|(q + 6) \in O(|V(G^0)|^2 q)$ vertices. Furthermore, for Δ being the maximum degree of H^0, we obtain the host graph H^2 having maximum degree at most $\Delta + 1$. Due to a result of Cai, Chan, and Chan [CCC06, Theorem 1], Subgraph Isomorphism is fixed-parameter tractable for the combined parameter "number of vertices of the pattern graph and maximum degree of the host graph," which yields the result. □

Theorem 4.14 mainly provides a classification result. It remains open to further improve the running time in order to obtain a practically relevant efficient algorithm. Theorem 4.14 concludes our results for C-sGEF-A. A compact overview of the results in the tabular form is depicted in Table 4.3.

4.5 Finding Strongly Graph-Envy-Free Allocations

We move on to the strong variant of our envy-freeness concept and analyze how this stronger notion affects the computational complexity (Table 4.5). Again, we start with those restrictions of C-sGEF-Allocation that result in the computationally least hard variants of the problem; specifically, we restrict the utility functions to be identical and the attention graph to be strongly connected. At the beginning, it might come a bit surprising that for C-sGEF-A the computationally simplest case is not the one of acyclic attention graphs (recall that the case of identical utility functions and strongly connected attention graphs was NP-hard for C-GEF-A, see Proposition 4.9). However, observing that as soon as the "greater than" relation forms a cycle, we arrive at a trivial

[2]The rows referring to more specific utility functions are omitted, as they are subsumed by row "additive."

attention graph type \ preferences type	parameterization				
	#agents	#resources	outdegree	#agents + outdegree	#resources + outdegree
directed acyclic					
additive[2]	P ♣	P ♣	P ♣	P ♣	P ♣
strongly connected					
id. 0/1	P ♡	P ♡	P ♡	P ♡	P ♡
id.	W[1]-h †	FPT ♠	p-NP-h †	W[1]-h †	FPT ♠
0/1	FPT §	W[1]-h ‡	p-NP-h ‡	FPT §	W[1]-h ‡
additive	W[1]-h †	W[1]-h ‡	p-NP-h †	W[1]-h †	W[1]-h ‡
general					
id. 0/1	FPT §	W[1]-h ◇	p-NP-h ◇	FPT §	FPT •
id.	W[1]-h †	W[1]-h ◇	p-NP-h ◇	W[1]-h †	FPT •
0/1	FPT §	W[1]-h ◇	p-NP-h ◇	FPT §	W[1]-h ‡
additive	W[1]-h †	W[1]-h ◇	p-NP-h ◇	W[1]-h †	W[1]-h ‡

Reference legend: ◇ Th. 4.7 † Pr. 4.9 ‡ Th. 4.11 § Pr. 4.8 ♣ Obs. 4.4 ♡ Cor. 4.6 ♠ Th. 4.10 • Th. 4.14

Table 4.3.: Parameterized complexity of C-GEF-Allocation. The results are grouped by three criteria: the attention graph type, the preference type, and the parameterization. All cases except for the polynomial-time solvable ones are NP-hard.

impossibility (assuming identical utility functions), immediately explains the situation. We present this observation formally below.

Observation 4.16. *Let $\mathcal{G}$ be a graph that contains a strongly connected component with more than one vertex. Then, there is no strongly graph-envy-free allocation if the agents have identical preferences.*

Proof. By definition, there is a cycle in every strongly connected graph with more than one vertex. Let us arbitrarily choose some agent a from the cycle. Let us call its predecessor a_-. Now, by the definition of strongly graph-envy-free allocation and transitivity of the "greater than" relation, we have that $u(\pi(a)) > u(\pi(a_-))$ and $u(\pi(a_-)) > u(\pi(a))$—a contradiction. □

Next, we present Algorithm 4.1 which, applying Observation 4.16, finds a complete, strongly graph-envy-free allocation for the case of identical 0/1 preferences and arbitrary input graphs.

Proposition 4.17. C-sGEF-Allocation *for identical* 0/1 *preferences can be solved in linear time.*

Procedure 4.1: Let $\mathcal{R}$ be a set of resources, let $\mathcal{A}$ be a set of agents such that every agent assigns the preference value of one to every resource, and let $\mathcal{G} = (\mathcal{A}, E)$ be a directed graph.

if $|\mathcal{A}| = 1$ **then**
  Allocate all resources to the single vertex; **return**
if *there exists a cycle in* $\mathcal{G}$ **then**
  No allocation is possible; **return**
Build a graph $G' = (\mathcal{A} \cup \{v_s\}, E')$ where $E' \coloneqq \{(u, v) \colon (v, u) \in E\} \cup \{(v_s, u) \colon u \in \mathcal{A} \wedge |N_G(u)| = 0\}$
Assign every vertex $w \in V$ a label $\ell(w)$ being the length of the longest path from v_s decreased by one
if $|\mathcal{R}| \geq \sum_{w \in W} \ell(w)$ **then**
  Assign $\ell(w)$ arbitrary resources from $\mathcal{R}$ to every agent $w \in V$
  Assign the remaining resources to arbitrary agents with zero in-degree in graph $\mathcal{G}$; **return**
No allocation is possible; **return**

Proof. By Observation 4.2, we know that we can assume without loss of generality that there are no resources with value zero. Hence, in Algorithm 4.1 we safely assume that every resource is assigned utility one by every agent.

Algorithm 4.1 first checks whether the graph either consists of only one vertex or contains a cycle. If the former is true, then it is enough to give it all the resources to obtain a complete and strongly graph-envy-free allocation. If the input graph is cyclic, then by Observation 4.16 no feasible allocation exists.

Giving resources to some agent with zero in-degree cannot break strong graph envy-freeness. Thus, the task reduces to finding a strongly graph-envy-free (possibly incomplete) allocation π that guarantees strong graph envy-freeness for all agents and then to distribute the remaining resources to agents with zero in-degree. An allocation π should, naturally, use as few resources as possible. Algorithm 4.1 finds such an allocation π building graph G' and assigning every agent $w \in V$ a label $\ell(w)$. Label $\ell(w)$ is the minimal number of resources that w has to get to achieve a strongly graph-envy-free allocation with the smallest number of resources. We make an inductive argument based on the label value assigned by the algorithm to prove this claim. Let us focus on the input graph $\mathcal{G}$. Since the agents in $\mathcal{A}$ with label 0 are sinks, it is clear that giving them no resources never violates strong graph envy-freeness. Let us consider some label

value $x > 0$ and an agent $a \in \mathcal{A}$ with $\ell(a) = x$. Because x is the length of a path from a to the furthest sink, there exists an arc (a, a') such that agent a' has label $x - 1$ and gets a bundle of at least $x - 1$ resources. Thus, indeed, agent a has to get at least x resources to achieve a strongly graph-envy-free allocation.

Using the breadth-first search, we can assign the labels to the agents and check whether a graph is cyclic in linear time. Since the same holds for our procedure of building auxiliary graph G', Algorithm 4.1 runs in linear time. □

The linear-time solvability settled in Proposition 4.17 heavily depends on the identical 0/1 preferences. Indeed, in the following Proposition 4.18 we show that C-sGEF-A becomes intractable for identical preferences in the case of acyclic attention graphs, which stands in contrast to C-GEF-Allocation that is always solvable in polynomial time if the attention graph is acyclic. Reducing from the NP-hard Unary Bin Packing [Jan+13], we mainly use the fact that in a (directed) path over k agents, the first agent has to get a bundle with utility at least $k - 1$.

Proposition 4.18. C-sGEF-Allocation *with identical monotonic additive preferences is* NP*-hard even if the input graph is acyclic and the maximal out-degree is one.*

Proof. We give a polynomial-time many-one reduction from Unary Bin Packing [Jan+13] where, for a given multiset of integer item sizes encoded in unary, a bin size b, and the maximal number of bins k, the question is whether it is possible to partition the items to at most k bins each with capacity b.

Let $I = (\mathcal{S}, b, k)$ be an instance of Unary Bin Packing, where $\mathcal{S} \coloneqq \{s_1, s_2, \ldots, s_n\}$ and $S \coloneqq \sum_{i=1}^{n} s_i$. Without loss of generality we assume that $S = k \cdot b$. We create an instance of sGEF-Allocation with the following input: The set $R \coloneqq \{r'_1, r'_2, \ldots, r'_b\} \cup \{r_1, r_2, \ldots, r_n\} \cup \{r^*_1, r^*_2, \ldots, r^*_k\}$ of resources, and the set $\mathcal{A}$ of agents, containing *bin agents* $\{a_1, a_2, \ldots, a_k\}$, *dummy agents* $\{a'_1, a'_2, \ldots, a'_b\}$, and k *special agents* $a^*_1, a^*_2, \ldots, a^*_k$. To form the graph G describing the agent relations we first build a (directed) path $(a'_b, a'_{b-1}, \ldots, a'_1)$ through the dummy agents. Then we create an arc from every bin agent to a'_b. Finally, for each $i \in [k]$, we create an arc from a^*_i to a_i. For $i \in [b]$, we set the value of each r'_i to be $i - 1$. For $i \in [n]$, we set the value of r_i to be s_i. The value of the special resources is set to $b + 1$.

According to graph G, for an allocation to be strongly graph-envy-free, each dummy agent a'_i, $i \in [b]$, has to get resources of total value at least $i - 1$. This implies that all bin agents have to achieve a utility of b, and each special agent

has to get resources valued at least $b+1$. This means that the minimal value of allocated resources is exactly $\sum_{i\in[b]}(i-1)+S+k(b+1)$. However, the sum of utilities of all resources is exactly the same. Hence, one can only allocate the resources achieving strong graph envy-freenessf one can allocate resources representing the items to pack $\mathcal{S}$ to bin agents.

The reduction is polynomial-time executable, thus proving NP-hardness. Clearly, no agent vertex has out-degree higher than one. □

Observe that the NP-hardness from Proposition 4.18 holds even if each agent can envy at most one other agent. On the positive side, C-sGEF-A with identical utility functions turns out to be fixed-parameter tractable with respect to the number of agents. In the following Proposition 4.19, we show this combining a brute-force approach with solving the ILP model of C-sGEF-A from Section 4.3.4.

Proposition 4.19. C-sGEF-Allocation *with identical preferences is fixed-parameter tractable with respect to the number of agents.*

Proof. A general approach in the proof is to distinguish two cases depending on whether there are more distinct utility values reported by agents than agents themselves. It turns out that if this is the case, then we can efficiently brute-force a given instance. Otherwise, we employ the ILP model from Section 4.3.4 and show that the number of variables is upper-bounded by the number of agents, obtaining fixed-parameter tractability due to Lenstra's result [Len83] recalled in Proposition 2.1.

Consider an instance of C-sGEF-A with an attention graph $\mathcal{G}$, a set $\mathcal{R}$ of resources, and the number u_{diff} of different utility values assigned to $\mathcal{R}$ by the agents of the instance. Since checking whether a graph is acyclic can be done in polynomial time, thanks to Observation 4.5, we can assume without loss of generality that $\mathcal{G}$ has no cycles.

Let us first consider the case in which $u_{\text{diff}} > |\mathcal{A}|$. Since $\mathcal{G}$ is a directed acyclic graph, it has a topological ordering $\succ$ computable in polynomial time. Furthermore, there clearly exists at least one agent a^* in $\mathcal{G}$ with no incoming arc. These observations allow us to find a strongly graph-envy-free allocation in two polynomial-time computable steps. First, using the fact that $u_{\text{diff}} > |\mathcal{A}|$, we select exactly $|\mathcal{A}|$ differently valued resources and distribute them such that if $a \succ a'$, then a gets a more valuable resource. The second step is to give all remaining resources to a^*. Because $\succ$ is a topological ordering, for every

arc (a, a') it holds that $a \succ a$. Since a has, by definition, a resource with greater value than that of a', a does not envy a'.

Let us now consider the case where $u_{\text{diff}} \leq |\mathcal{A}|$. In the ILP model from Section 4.3.4 the number of variables is the product of the number of agents and the number of different possible resource types. Recall that a type of some resource is defined as a vector of utility values given to the resource by all agents. Thus, in our case of identical utility functions, a resource type boils down to a single number that is the utility given by the agents to a particular resource. By the assumption of this case, the number of different utilities given to the resources by agents is upper-bounded by $|\mathcal{A}|$. Eventually, the whole number of variables in the ILP model from Section 4.3.4 is upper-bounded by $|\mathcal{A}|$ which gives us fixed-parameter tractability by Lenstra's result [Len83] from Proposition 2.1. □

Continuing good news brought by Proposition 4.19, we provide a fairly general positive result for the case with few resources. Specifically, in Theorem 4.20, we show that C-sGEF-A is fixed-parameter tractable with respect to the number of resources, independently of the attention graph structure and the preference type.

Theorem 4.20. C-sGEF-Allocation *is fixed-parameter tractable with respect to the number of resources.*

Proof. A high-level idea in the proof is to identify certain special cases of C-sGEF-A depending on whether the structure of the input graph. Then, for each case, we use a different way of solving the case, exploiting features that that the case displays.

We can divide vertices of every directed graph into three groups. Group $V_{\mathtt{s}}$ consists of *sources*, that is vertices with in-degree zero. Group $V_{\mathtt{t}}$ consists of *sinks*, that is vertices with out-degree zero. All other vertices belong to group $V_{\mathtt{i}}$. Observe that no vertex belongs to $V_{\mathtt{s}} \cap V_{\mathtt{t}} \cap V_{\mathtt{i}}$ but there might be vertices that belong to $V_{\mathtt{t}} \cap V_{\mathtt{s}}$.

Let us denote the number of agents, which are vertices, by n and the number of resources by m. We distinguish different cases and for each them provide an algorithm to solve it:

1. $m \geq n$. We check all possible allocations of the resources to the agents. Since there are at most n^m possible allocations, by the assumption that there are at least as many resources as agents, we obtain that the number of possible allocations is upper-bounded by m^m.

2. $m < |(V_s \cup V_i) \setminus V_t|$. There is no strongly graph-envy-free allocation because each agent from $(V_s \cup V_i) \setminus V_t$ has to get at least one resource.

3. $m = |(V_s \cup V_i) \setminus V_t|$. We check all possible, at most $O(m!)$ allocations.

4. $|(V_s \cup V_i) \setminus V_t| < m < n$ and $V_s \neq \emptyset$. Since we have at least one source, that can get arbitrarily many resources, there always is a strongly graph-envy-free allocation in which no sink agent gets any of the resources. Hence, one can ignore the sink agents and check all $|(V_s \cup V_i) \setminus V_t|^m$ allocations in FPT-time (with respect to m, as in Case 1).

5. $|(V_s \cup V_i) \setminus V_t| < m < n$ and $V_s = \emptyset$. First, observe that when $V_s = \emptyset$, then, the condition for this case gives us that $|V_i| < m$ (recall that $V_i \cap V_t = \emptyset$ by definition). In this case, unlike in Case 4, we cannot easily ignore the sink agents because there might be scenarios in which these agents have to get some resources. Consider some sink agent $t \in V_t$ and let $N(t)$ be the set of agents from which there is an arc to t. We want to describe all sinks $t \in V_t$ by their respective sets $N(t)$; hence, for some sink $t \in V_t$, we call the set $N(t)$ a *type* of t. Observe that there are at most $2^{|V_i|} < 2^m$ different types of sink vertices. This means that without exceeding the FPT-time, we can guess which resources will be allocated to each agent in V_i and which resources will be allocated to the sink agents of a certain type (there are, respectively, at most $O(m)$ and $O(m^{2^m})$ such guesses). Clearly, if there is a strongly graph-envy-free allocation, then it is described by such a guess. To show how to proceed with a guess, let us fix an arbitrary one. We show that, using this guess, we are able to, again, guess a strongly graph-envy-free allocation (if it exists) in FPT-time. The clue is to correctly allocate the guessed resources to the given sets of sink vertices of different types. Observe that for all sink agents of the same type we can ignore their utility functions, as the sink agents cannot envy (they have no outgoing arcs). Moreover, all sink agents of the same type have incoming arcs from exactly the same inner vertices, which makes the sink agents indistinguishable. Furthermore, there are at most m resources and thus, we have to distribute the guessed resources for the sink agents of the particular type to at most m of these sink agents. As a result, for each type, we can check all possible allocations of the guessed resources for this type to at most m arbitrarily selected sink agents of this type. There are at most 2^m types and for each of them we have at most $O(m^m)$ possible guesses of how to distribute resources within this type.

The list of above cases is exhaustive. Indeed if $m \geq n$, then we obtain the first case; otherwise, we list all possible cases with respect to the relation between m and $|(V_s \cup V_i) \setminus V_t| < n$ (with one case split to two depending on whether the set V_s is empty or not). □

Propositions 4.18 and 4.19 are tailored to the case of identical preferences. Thus, they do not cover the case of 0/1 preferences without the same utility functions. However, the following Proposition 4.21, using a reduction from Clique, shows that C-sGEF-A remains hard also in the case of 0/1 preferences. The result holds even for acyclic attention graphs.

Proposition 4.21. C-sGEF-Allocation *with* 0/1 *preferences is* NP*-hard for an input graph being either strongly connected or acyclic even if the maximal out-degree is three.*

Proof. We prove Proposition 4.21 by giving a polynomial-time many-one reduction from Clique. We first present the reduction for C-sGEF-A with an acyclic attention graph. Then we massage the reduction to cover the case of an attention graph being strongly connected.

In the Clique problem, for a given graph and an integer k, we ask whether there is a set of k pairwise adjacent vertices in the given graph. Let I be a Clique instance formed by an undirected graph $G = (V, E)$ with a set $V = \{v_1, v_2, \ldots, v_n\}$ of vertices and a set $E = \{e_1, e_2, \ldots, e_m\}$ of edges, and a target clique size k. Without loss of generality, assume that $1 < k < n$ and $m > \binom{k}{2}$.

We construct an instance I' of C-sGEF-A associating each vertex and edge of G with an agent, adding n *separating agents*, and the source s and sink t agents. Formally, we set $\mathcal{A} := V \cup E \cup \{v_1^*, v_2^*, \ldots, v_n^*\} \cup \{s, t\}$. The following steps describe the construction of graph $\mathcal{G}$; the construction is depicted in Figure 4.6.

1. For each edge $e = \{v, v'\} \in G$, we add three arcs, (v, e), (v', e), and (e, t), to $\mathcal{G}$.
2. For each vertex $v_i \in V$, $i \in [|V|]$, we add arc (v_i^*, v_i) to $\mathcal{G}$.
3. For each vertex $v_i \in V$, $i \in [|V| - 1]$, we add arc (v_i, v_{i+1}^*) to $\mathcal{G}$.
4. We add arc (s, v_1^*).

Then, we add m *edge resources* and $n + k$ *vertex resources*. We arbitrarily split the edge resources into two sets: set $\mathcal{R}^-$ of $m - \binom{k}{2}$ edge resources and set $\mathcal{R}^+$ of $\binom{k}{2}$ edge resources. We refer to the latter ones as *distinguished edge resources*.

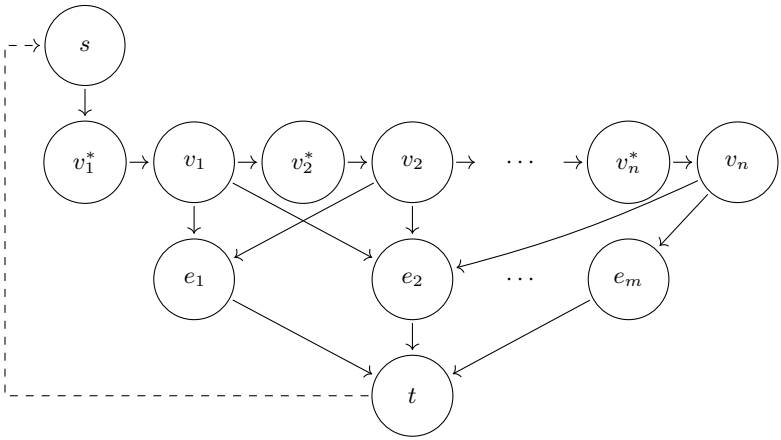


Figure 4.6.: The construction in the proof of Proposition 4.21. The dashed arc is used in the case of the input graph being strongly connected.

We denote the set of vertex resources by $\mathcal{R}^{\bullet}$. Furthermore, we add n *separating resources*, using $\mathcal{R}^{\shortparallel}$ to refer to them, and a *special resource* $r^{\triangle}$. We define the utilities of the resources as depicted in Table 4.4. Naturally, the described construction of a single instance of C-sGEF-A is computable in polynomial time.

To show the correctness of the above reduction and finish the proof, we state the following useful lemma, which gives the requirements of graph-envy-free allocations for instance I'.

Lemma 4.22. *In an allocation π for instance I', the vertex agents, the edge agents and agent s are not envious if and only if*

1. $r^{\triangle} \in \pi(s)$,
2. $\forall i \in [n]\colon |\pi(v_i^*) \cap \mathcal{R}^{\shortparallel}| = 1$, *and*
3. $\forall e \in E\colon |\pi(e) \cap (\mathcal{R}^{-} \cup \mathcal{R}^{+})| = 1$.

Proof. We prove the lemma for both directions separately.

($\Rightarrow$) Claim 1 holds because agent s, having an outgoing arc, gives a positive utility only to resource $r^{\triangle}$. Hence, s has to get $r^{\triangle}$. Similarly, there is the same

	s	V	E	t	$v_1^*, \ldots v_n^*$
$r^{\triangle}$	1	0	0	0	0
$\mathcal{R}^{-}$	0	0	1	0	0
$\mathcal{R}^{+}$	0	1	1	0	0
$\mathcal{R}^{\parallel}$	0	0	0	0	1
$\mathcal{R}^{\bullet}$	0	1	0	0	0
$r^{\triangledown}$	0	0	0	1	0

Table 4.4.: Utilities of resources in the proof of Proposition 4.21. Resource $r^{\triangledown}$ is needed only to prove Proposition 4.21 for the input graph being strongly connected.

number of separating resources and separating agents, who have an outgoing arc each. The separating resources are the only resources to which the separating agents assign positive utility; thus, each separating agent has to get a separating resource, which is exactly what Claim 2 formalizes. The very same argument for the edge agents yields Claim 3.

($\Leftarrow$) Reusing the argumentation from the opposite direction, it is immediate that allocation π does not introduce envy if we neglect the vertex resources. However, observe that the vertex resources are given utility zero by every agent except for the vertex agents. Thus, no matter how we allocate the vertex resources, all non-vertex agents remain unenvious. □

In other words, Lemma 4.22 says that each complete strongly graph-envy-free allocation for I' gives all separating resources to the separating agent, one resource per agent; gives all edge resources to the edge agents, one resource per agent; and gives $r^{\triangle}$ to agent s. (Note that Lemma 4.22 does not specify how to allocate the vertex resources.) Using this convenient (partial) characterization of solutions to I', in the following we prove correctness of the presented reduction.

Suppose $C = (V_C, E_C)$ is a clique of size k (i.e., consisting of k vertices) in G. We construct a complete strongly graph-envy-free allocation π for instance I'. We allocate the special resource to s, the separating resources to the separating agents (a resource per agent), and the edge resources to the edge agents such that each $e \in E_C$ gets a distinguished edge resource. To each agent $v \in V$, we allocate two vertex resources if v belongs to V_C, or else we allocate one vertex resource to v. Clearly, all resources are allocated. Thanks to Lemma 4.22, it remains to show that no vertex agent is envious. Observe that if $v \in V_C$, then v values its bundle at two. As a result, v cannot envy because the only arcs it has are arcs pointing to some edge agents and each of them has a single resource.

So, towards a contradiction, let $v \in V \setminus V_C$ be an envious vertex agent. Since v gets one resource that v values at one, v can only envy an edge agent with a distinguished resource. Thus, let $e \in E_C$ be such an agent to which v is pointing. This means that edge e belongs to clique C but v does not belong to this clique; a contradiction.

For the reverse direction suppose π is a solution to I'. Due to Lemma 4.22, we know that there are exactly $\binom{k}{2}$ edge agents with a distinguished resource; we refer to them as *selected edge agents*. By construction, π has to give each vertex agent at least one vertex resource and every vertex agent that is pointing to a selected edge agent two vertex resources; we call the latter *selected vertex agents*. To avoid envy, each selected edge agent can only have an incoming arc from two selected vertex agents. Associating selected vertex agents with vertices and selected edge agents with edges, this situation exactly maps to finding a clique in graph G. This concludes the proof for $\mathcal{G}$ being acyclic.

We can easily adapt the aforementioned reduction to the case where the attention graph is strongly connected. We do so by adding an arc (t, s) and one special resource $r^\triangledown$. We let only agent t give utility one to $r^\triangledown$. We claim that these two problems are equivalent; that is, there is a one-to-one mapping between complete strongly graph-envy-free allocations for them.

Let π be a strongly graph-envy-free allocation for an acyclic attention graph $\mathcal{G}$. We obtain an allocation π' for a strongly connected graph $\mathcal{G}'$ by copying π and additionally giving $r^\triangledown$ to t. Clearly, the only envy that could have emerged, between t and s, is prevented by $r^\triangledown$. Since every agent except for t gives $r^\triangledown$ utility zero, resource $r^\triangledown$ cannot change the envy state of another agents.

Conversely, if an allocation π' is strongly graph-envy-free for $\mathcal{G}'$, then $\pi'(t) = \{r^\triangledown\}$. Thus, we construct an allocation π for the corresponding acyclic attention graph $\mathcal{G}$ by copying π' and removing resource $r^\triangledown$. Since arc (t, s) does not exist in $\mathcal{G}$, agent t, that gets no resource under π, is not envious under π. Naturally, every other agent was not envious under π' so it cannot be envious under π. □

Similarly as in the case of identical preferences, NP-hardness from Proposition 4.21 can be successfully tackled for the case of few agents of resources. Parameterized tractability of C-sGEF-A with respect to the number of resources has already been shown for a general case in Theorem 4.20. The following Corollary 4.23, which is a straightforward consequence of Proposition 4.8, complements the picture by stating that C-sGEF-A with identical utility functions is tractable for few agents.

Corollary 4.23. C-sGEF-Allocation *with* 0/1 *preferences is fixed-parameter tractable with respect to the parameter "number of agents."*

The last remaining question in our analysis of the computational hardness of C-sGEF-A is the parameterized computational complexity of C-sGEF-A with respect to the number of agents in the case of general monotonic additive utility functions. Theorem 4.24 provides a negative answer showing that for this parameterization C-sGEF-A remains intractable even for acyclic attention graphs.

Theorem 4.24. C-sGEF-Allocation *for monotonic additive preferences is* NP*-hard and* W[1]*-hard when parameterized by the number of agents even if the input graph $\mathcal{G}$ is either a directed path or a cycle.*

Proof. We give a parameterized reduction from the NP-hard Unary Bin Packing where, for a given collection of integer item sizes encoded in unary, the maximal number k of bins each of size B, the question is whether it is possible to partition the items to at most k bins respecting bin size B. Unary Bin Packing is known to be W[1]-hard with respect to the number of bins [Jan+13].

A general idea is to create a resource for each item and construct an agent to represent every bin. Each of the constructed agents gives every resource the same utility as the size of an item the resource represents. Then, we construct an instance of sGEF-Allocation so that every agent has to get a bundle to which it assigns utility at most B.

To present the reduction formally let us fix an instance $I = \{\mathcal{S}, B, k\}$ of Unary Bin Packing, where $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ and $S = \sum_{i=1}^{n} s_i$. Without loss of generality, we assume that $S = k \cdot B$. We create a new instance of sGEF-Allocation with the following input. Let $\mathcal{R} \coloneqq \{r_1, \ldots, r_n, r_{n+1}, r_{n+2}, \ldots, r_{n+k}, r^*\}$ be the set of resources and let $\mathcal{A} \coloneqq \{a_1, a_2, \ldots, a_{k+2}\}$ of agents. Associating vertices with agents, we construct graph $\mathcal{G}$ which is a directed path $(a_1, a_2, \ldots, a_{k+2})$. Agent a_{k+2} gives no utility to every resource. Moreover resource r^* is assigned a non-zero utility only by the two agents a_k and a_{k+1}, that is, $u_{k+1}(r^*) \coloneqq 1$, $u_k(r^*) \coloneqq \frac{S}{k} = B$ and $\forall_{i \in [k]} u_i(r^*) \coloneqq 0$. For every resource r_i, $0 < i \leq n$, we set the utility function values to s_i for agents from $\{u_1, u_2, \ldots, u_k\}$ and zero otherwise. Resource r_{n+j}, $0 < j \leq k$, is assigned utility zero by all agents except for a_j, which assigns utility one.

To show that solving the new instance is equivalent to solving the initial instance of Unary Bin Packing, we first observe that agent a_{k+1} must have a non-zero utility. The only resource the agent can get to achieve this is r^*.

Observe that this implies that a_k has a bundle of utility at least $B + 1$. By obtaining the resources $r_{n+1}, r_{n+2}, \ldots, r_{n+k}$, agent a_k can get at most utility one; namely, from resource r_{n+k}. As a result, a_k has to get utility at least B by obtaining a selection of the resources $r_1, r_2, \ldots, r_n$. We can apply this argument iteratively, for the remaining agents $a_1, a_2, \ldots, a_{k-1}$. Thus, we get that each agent a_1 to a_k obtains a subset of $\{r_1, r_2, \ldots, r_n\}$ that it values to at least B. Since there are exactly k of these agents, this bound is tight. To fulfill the requirements of strong graph envy-freeness, each agent $a_i \in \{a_1, a_2, \ldots, a_k\}$ also gets resource r_{n+i}. The solution to Unary Bin Packing is now formed by sets of elements s_i corresponding to the bundles of agents $a_1, a_2, \ldots, a_k$.

So far we have shown the proof for the case of directed paths. However, one can add an arc from a_{k+1} to a_1, transforming the path into a cycle. Then, adding one resource liked only by agent a_{k+1} yields a proof for the case of a cycle.

Our reduction is executable in polynomial time. There are polynomially many agents with respect to the parameter k of Unary Bin Packing, which proves the theorem. □

We end this section summing up our analysis of the (parameterized) computational complexity of C-sGEF-A with Table 4.5.

4.6 Conclusion

We introduced a new model in the area of indivisible resource allocations by combining social networks with the classical notion of envy-freeness. Our model seems to be a promising line of (future) research (recently also taken up by Eduard et al. [Edu+20] and Lange and Rothe [LR19]). Our computational complexity analysis provided a number of parameterized computational hardness lower bounds. However, we came up with several parameterized tractability results and a few polynomial-time algorithms for a couple of specific scenarios. See Figure 4.7 for a compact, graphical representation of our findings. Note that our model clearly differs from the one of Chevaleyre, Endriss, and Maudet [CEM17], which considers distributed resource allocations of indivisible resources with monetary payments, which do not appear in model. In fact, introducing them to our considerations would have a huge impact on the results we obtained.

[3]The results for identical 0/1 preferences in this case are subsumed by the results presented in row "id."

attention graph type	preferences type	parameterization				
		#agents	#resources	outdegree	#agents + outdegree	#resources + outdegree
directed acyclic						
	id. 0/1	P ◇	P ◇	P ◇	P ◇	P ◇
	id.	FPT ♣	FPT ‡	p-NP-h §	FPT ♣	FPT ‡
	0/1	FPT ♠	FPT ‡	p-NP-h ♡	FPT ♠	FPT ‡
	additive	W[1]-h †	FPT ‡	p-NP-h †	W[1]-h †	FPT ‡
strongly connected[3]						
	id.	$O(1)$ •	$O(1)$ ()	$O(1)$ •	$O(1)$ •	$O(1)$ •
	0/1	FPT ♠	FPT ‡	p-NP-h ♡	FPT ♠	FPT ‡
	additive	W[1]-h †	FPT ‡	p-NP-h †	W[1]-h †	FPT ‡
general						
	id. 0/1	P ◇	P ◇	P ◇	P ◇	P ◇
	id.	FPT ♣	FPT ‡	p-NP-h §	FPT ♣	FPT ‡
	0/1	FPT ♠	FPT ‡	p-NP-h ♡	FPT ♠	FPT ‡
	additive	W[1]-h †	FPT ‡	p-NP-h †	W[1]-h †	FPT ‡

Reference legend: ◇ Pr. 4.17 † Th. 4.24 ‡ Th. 4.20 § Pr. 4.18 ♣ Pr. 4.19 ♡ Pr. 4.21 ♠ Cor. 4.23 • Obs. 4.16

Table 4.5.: Parameterized complexity of C-sGEF-Allocation. The results are grouped by three criteria: the attention graph type, the preference type, and the parameterization. All hardness results also imply classical NP-hardness.

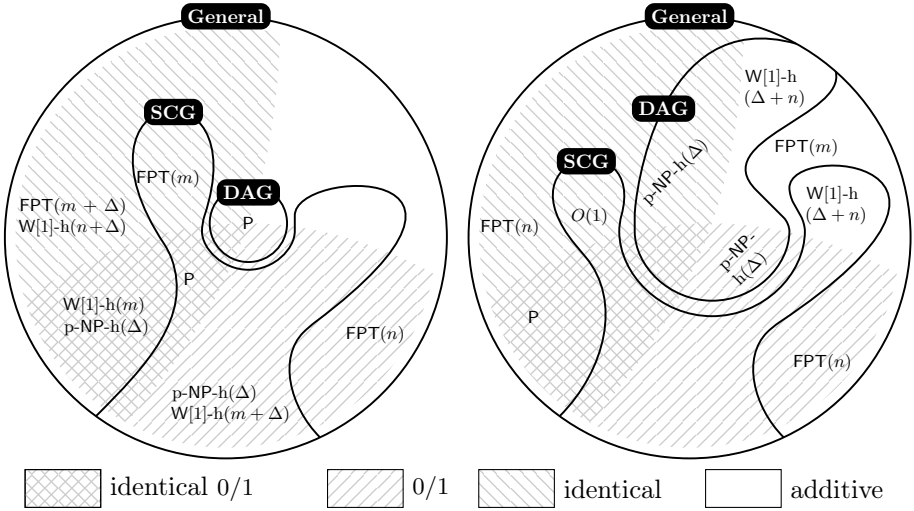


Figure 4.7.: A compact illustration of computational complexity of C-GEF-A(left) and C-sGEF-A(right), presented also in Tables 4.3 and 4.5. Sets represent spaces of all possible problem instances with a particular type of preferences and a particular structure of the attention graph. Types of preferences are indicated by the sets marked with background patterns. Different structures of the attention graph are indicated by the blobs tagged with labels. For example, an entry "P" in the set labeled as "SCG" and on checkered background in the left picture, means that C-GEF-A is in P when the attention graph is strongly connected (indicated by set "SCG") and the preferences are identical 0/1 (represented by the checkered background). Similarly, an entry "p-NP-hard" in the set labeled "DAG" on the left-to-right diagonal background on the right picture indicates that C-sGEF-A is para-NP-hard with respect to parameter Δ already when the attention graph is acyclic and directed (indicated by set "DAG"), and when the utility functions are 0/1. Naturally, this hardness transfers, for example, to C-sGEF-A for the case of general graphs and general utility functions.

In our study (especially for C-GEF-A), we presented a number of (parameterized) computational hardness results. In a sense, they lay the foundations for a more refined search for islands of tractability concerning practically motivated use cases of our basic models. To this end, there are plenty of opportunities. First, it appears natural to deepen our studies by considering various special graph classes for the underlying social network (besides classes of graphs with bounded tree-width or bounded clique-width studied by Eduard et al. [Edu+20]). Note, however, that the class of constant-degree graphs alone is not enough to achieve (fixed-parameter) tractability. Thus, one might need to combine parameters related to the underlying social network with such natural parameters as the maximum utility value, the number of resources per bundle or the number of different "types" of resources in order to achieve fixed-parameter tractability results. Our results also show that the aforementioned natural parameters alone also do not usually admit fixed-parameter tractability (e.g., parameter "maximum utility value" does not guarantee fixed-parameter tractability). In addition, one may move from directed to undirected graphs or one may consider graphs that only consist of small connected components. Finally, including further fairness and efficiency concepts, beyond the ones we studied, appears to be promising as well (see the conference paper behind this chapter [BKN18] for a preliminary study towards this direction). We demonstrate a detailed study on parameterization by the number of agents and the maximum utility value in the following Chapter 5, where we show that it leads to fixed-parameter tractability for a number of different variants of indivisible fair allocations problems.

We also provide several combinatorial algorithms obtaining fixed-parameter tractability of various variants of the problems we study. However, our theoretical results only provide the worst-case computational complexity. Thus, a natural follow-up is to experimentally assess the running times of our algorithms. Not only can such a study verify practical applicability of the algorithms, but it can also reveal important structural features of the studied problem instances. These insights could lead to improving the algorithms and lowering their running times. For example in Chapter 5, we take such an experimental approach and apply the algorithms devised therein to solve real-world instances.

The above paragraph brings up yet another challenge related to our model: to collect compatible real-world data. This could open the way for experimental, qualitative study of fair allocation in social context. Possibly, we could also answer such questions as "How likely is it in practice that an envy-free allocation respecting social relations can be found quickly?" or "How many graph-envy-free allocations are there on average?".

CHAPTER 5

High Multiplicity Allocations

In this chapter, we further study the parameterized computational complexity of problems in the context of fair allocations of indivisible goods. More specifically, we show fixed-parameter tractability results for a broad set of problems concerned with envy-free and Pareto-efficient allocations of resources (with agent-specific utility functions) to agents. In principle, this implies efficient exact algorithms for these (generally computationally intractable) problems whenever we face instances with few agents and low maximum (absolute) utility values. This holds true also in high-multiplicity settings where we may experience numerous copies of a single resource (modeled by encoding resource multiplicities in binary).

On the technical side, our approach provides an algorithmic theorem covering a number of fair allocation problems in the additive preferences model. To achieve this, our main technical contribution is to elaborately use tools from integer linear programming. More specifically, we exploit results originally going back to a famous result of Lenstra [Len83] (recalled in Proposition 2.1) concerning the fixed-parameter tractability of Integer Linear Programming with respect to the number of variables and the more recent framework of (Combinatorial) N-Fold Integer Programming [HKW10, HOR13]. We reveal and exploit a fruitful interaction between these two cornerstones in the theory of integer (linear) programming, which may be of independent interest in applications going beyond the fair allocation domain. We also experimentally assess running times of our algorithms on real-world data obtained from the authors of spliddit.org [Pro+20]. The achieved results provide strong evidence that our techniques are practically relevant.

5.1 Introduction

We continue seeking efficient ways of computing fair allocations by focusing on deriving fixed-parameter tractability results for parameters assumed to be small in relevant practical application scenarios. Using Integer Linear Programming methods, we succeed in providing a collection of fixed-parameter tractability results for natural problem parameters.

As introduced in Chapter 3, we focus on the model where agents evaluate resources (which are to be distributed among them) by individual utility functions, assigning each resource an agent-specific utility value. The quality of an allocation is determined by the sums of the utility values of the resources the agents received. The goal is to find an allocation that is fair to the agents. We consider an allocation to be fair when it is envy-free, that is, there is no agent wishing to swap its own bundle with a bundle of another agent (recall Definition 3.4 for a formal statement). To make this concept truly meaningful, usually *Pareto-efficiency* of an allocation is requested in addition to envy-freeness. Intuitively, an allocation is Pareto-efficient if there exists no allocation *dominating* it; that is, there is no other allocation such that at least one agent gets a bundle it values more than in the old allocation and all other agents get bundles with values at least as good as they had before. However, note that an envy-free and Pareto-efficient allocation sometimes does not exist (for instance, consider the trivial case of two agents and one resource).

Next, let us become more specific about the concrete allocation problems we study with respect to fairness and Pareto-efficiency. As said, due to the general computational hardness, we study more constrained scenarios, reflected by choosing some problem-specific parameters in the spirit of parameterized complexity analysis. The two most central parameters for our studies are

- the number of agents (there even are prominent resource allocation protocols for two agents, so it is realistic to assume that in many relevant applications this is a small number) and
- the maximum absolute utility value (usually high maximum utility implies a lot of different possible utility values, and for humans it is often difficult to distinguish between too many of them [Bot+04]; so, a small maximal absolute utility value is usually a realistic assumption).

We will allow for high-multiplicity resources, making it possible to compactly encode large input instances. In addition, we will provide results for a number of relaxations of envy-freeness and Pareto-efficiency. One of such relaxations is envy-freeness up to one good (EF1) [Lip+04]. Here, every agent a is envious if there is another agent b with whom a would prefer to swap its bundle even if a would ignore the most valuable resource in b's bundle. Further concepts include: envy-freeness up to any good (EFX) [Car+16, PR18] and graph-envy-freeness discussed in Chapter 4. Amanatidis, Birmpas, and Markakis [ABM18] provide a comparison of approximate or relaxed fairness notions.

utilities (resources)	**w**hiteboard	**l**aptop	**c**ompute server	**p**rogramming student	**m**ath student
theory group	3	2	1	-1	3
a. i. group	1	2	4	3	2
software group	1	3	1	3	-1

Figure 5.1.: Utilities that the three research groups from the toy example report for the tools.

Roughly speaking, our work provides fairly universal, Integer Linear Programming-based tools that provide a number of fixed-parameter tractability results exploiting the parameters number of agents and maximum absolute value of utility. Specifically, we show tractability of finding envy-free and Pareto-efficient allocations for several variants of envy-freeness. Doing so, besides significantly extending the range of known fixed-parameter tractability results in this context, we also provide useful integer programming-based techniques that may be of independent interest beyond applications in the context of resource allocations. In particular, as a by-product we resolve two cases left open by Bliem, Bredereck, and Niedermeier [BBN16]: we show that computing an envy-free and Pareto-efficient allocation of indivisible goods is fixed-parameter tractable for the combined parameter number n of agents plus the maximum value $u_{\max}$ of utility for binary encoded additive preferences. (We complete their Table 1 [BBN16] for the slightly weaker parameter combination $n + u_{\max}$ but leave the complexity for $n + z$, where z denotes the number of different utility values, open.)

5.1.1 An Illustrative Example of Fair Allocations

Before more formally defining the investigated computational problems and presenting our corresponding main results, we present a toy example motivating and helping to understand the subsequently presented results. Assume that there are four whiteboards (**w**), ten laptops (**l**), four compute servers (**c**), two programming students (**p**), and one math student (**m**) to be allocated to three research groups in the computer science department: theory group (**t**), artificial intelligence group (**a**), and software group (**s**). Furthermore, assume that the utilities of the research groups for these resources can be expressed numerically as depicted in Figure 5.1. Observe that while tools usually have a non-negative utility, supervising a student might come even with negative utility, for example, when the student does not fit well to the group.

allocation π_1	
t	$4 \times \mathbf{w}$, $1 \times \mathbf{m}$
a	$4 \times \mathbf{c}$, $1 \times \mathbf{p}$
s	$10 \times \mathbf{l}$, $1 \times \mathbf{p}$

$u(\pi_1)$	**t**	**a**	**s**
t	15	3	19
a	6	19	23
s	3	7	33

allocation π_2	
t	$4 \times \mathbf{w}$, $1 \times \mathbf{l}$, $1 \times \mathbf{m}$
a	$4 \times \mathbf{c}$, $2 \times \mathbf{p}$
s	$9 \times \mathbf{l}$

$u(\pi_2)$	**t**	**a**	**s**
t	17	2	18
a	8	22	18
s	6	10	27

allocation π_3	
t	$4 \times \mathbf{w}$, $2 \times \mathbf{l}$, $1 \times \mathbf{m}$
a	$4 \times \mathbf{c}$, $2 \times \mathbf{p}$
s	$8 \times \mathbf{l}$

$u(\pi_3)$	**t**	**a**	**s**
t	19	2	16
a	10	22	16
s	9	10	24

allocation π_4	
t	$4 \times \mathbf{w}$, $1 \times \mathbf{l}$, $1 \times \mathbf{m}$
a	$4 \times \mathbf{c}$, $1 \times \mathbf{l}$, $1 \times \mathbf{p}$
s	$8 \times \mathbf{l}$, $1 \times \mathbf{p}$

$u(\pi_4)$	**t**	**a**	**s**
t	17	5	15
a	8	21	19
s	6	10	27

Figure 5.2.: Different allocations illustrating several aspects of our studies on envy-free Pareto-efficient allocations.

An allocation that maximizes the social welfare, that is, allocation π_1 in Figure 5.2, simply allocates every resource to one of the agents that valuates it the most. For instance, the theory group will get all whiteboards and the math student, the artificial intelligence group will get all compute servers and a programming student, and the software group will get all laptops and a programming student. As we can see, however, in table $u(\pi_1)$, where we list in each table entry the utility value of the bundle of resources allocated to the "column agent" viewed from the perspective of the respective row agent, the theory group as well as the artificial intelligence group envy the software group. Thus, allocation π_1 is Pareto-efficient but not envy-free.

To obtain an envy-free allocation we can, starting with π_1, transfer resources from the software group to the artificial intelligence group and the theory group. For example, the theory group would be unenvious by getting one laptop and the artificial intelligence group would be unenvious by getting the second programming student. However, if we perform both transfers simultaneously, then we end up with allocation π_2 which is still not envy-free since the theory group now still envies the software group. One can transfer one more laptop from the software group to the theory group to obtain the envy-free and Pareto-efficient allocation π_3.

Notably, just transferring one laptop from the software group to the theory group and one laptop from the software group to the artificial intelligence group also leads to an envy-free allocation (see π_4). However, this allocation is not Pareto-efficient because it is "dominated" by π_2.

5.2 Preliminaries

For a matrix $A \in \mathbb{Z}^{m \times n}$, the ℓ_1-norm $\|A\|_1$ is the sum of the absolute values of all of its entries, that is, $\|A\|_1 \coloneqq \sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij}|$; the ℓ_∞-norm $\|A\|_\infty$ is defined in a similar way (by changing the sums with maxima): $\|A\|_\infty \coloneqq \max_{i \in [n]} \max_{j \in [m]} |a_{ij}|$.

5.2.1 High-Multiplicity Allocations and Changes

Consider an arbitrary set $\mathcal{A}$ of n agents and a set $\mathcal{R}$ of resources among which some come in multiple copies; for example, there might be a bunch of identical chocolate bars or identical computers. As Definition 3.1 states, an allocation is a function that assigns the resources to the agents. In the presented case, however, it might be more convenient to look at an allocation as a function that assigns a certain number of resources of each kind to each agent. The following

definitions capture this concept of *high-multiplicity* resources formally.

Let $\mathcal{R}$ be a collection of resources of h different types, where for each type $t \in \mathcal{T} \coloneqq [h]$ there are $\#(t)$ copies of the resource of type t. Consequently, one can naturally encode an allocation using an integral vector of dimension $n \cdot h$, leading to the following alternative definition of an allocation.

Definition 5.1. An *allocation* $\boldsymbol{\pi} \in \mathbb{N}^{nh}$ is a vector $\left(\pi^1_{a_1}, \ldots, \pi^1_{a_n}, \pi^2_{a_1}, \ldots, \pi^h_{a_n}\right)$, where agent a_i is allocated $\pi^t_{a_i}$ resources of type $t \in \mathcal{T}$.

Alternatively, we could have defined an allocation as a two dimensional $h \times n$ matrix. However, in our application in which we frequently use integer linear programs with allocations as variables, a vector representation is more convenient and matches the definition of the ILP problem from Section 2.6 (where all variables are contained in a vector).

An allocation may be transformed into a different allocation $\hat{\boldsymbol{\pi}}$ using an *exchange*.

Definition 5.2. Let $\boldsymbol{\pi} \in \mathbb{N}^{nh}$ be an allocation. An *exchange* vector $\boldsymbol{\eta} \in \mathbb{N}^{n^2 h}$ is a non-negative integral vector, that is, $\boldsymbol{\eta} = \left(\eta^t_{a \to a'}\right)_{t \in \mathcal{T}, a, a' \in \mathcal{A}}$ such that, for every resource type $t \in \mathcal{T}$, it holds that $\sum_{a \in \mathcal{A}} \sum_{a' \in \mathcal{A}} \eta^t_{a \to a'} \leq \#(t)$.

Intuitively, an exchange $\boldsymbol{\eta} = \left(\eta^t_{a \to a'}\right)_{t \in \mathcal{T}, a, a' \in \mathcal{A}}$ should be understood in a way that agent a gives $\eta^t_{a \to a'}$ many resources of type t to agent a', as demonstrated in Example 5.1. The additional requirement in Definition 5.2 is a sanity check that an exchange leaves the total number of resources intact.

> **Example 5.1.** Consider two types t_1 and t_2 of resources, each having two copies. Furthermore, consider two agents a and b, and an allocation $\boldsymbol{\pi} = \left(\pi^{t_1}_a, \pi^{t_1}_b, \pi^{t_2}_a, \pi^{t_2}_b\right) = (2, 0, 0, 2)$. Suppose that agent a gives one resource of type t_1 to b and agent b gives one resource of type t_2 to a. This swap can be modeled by exchange $\left(\eta^1_{a \to b}, \eta^1_{b \to a}, \eta^2_{a \to b}, \eta^2_{b \to a}\right) = (1, 0, 0, 1)$ and results in an allocation $\hat{\boldsymbol{\pi}} = \left(\hat{\pi}^{t_1}_a, \hat{\pi}^{t_1}_b, \hat{\pi}^{t_2}_a, \hat{\pi}^{t_2}_b\right) = (1, 1, 1, 1)$.

Based on the so-called moves in societies introduced by Knop, Koutecký, and Mnich [KKM18], we now proceed with further definitions capturing the dynamics of allocations. Observe that a given exchange can be implemented for some allocations, whereas it might be impossible to do for other allocations.

Definition 5.3. For a given allocation $\boldsymbol{\pi}$, an exchange $\boldsymbol{\eta} = \left(\eta^t_{a\to a'}\right)_{t\in\mathcal{T},a,a'\in\mathcal{A}}$ is *admissible* if

$$\pi^t_a + \sum_{a'\in\mathcal{A}} \eta^t_{a'\to a} - \sum_{a'\in\mathcal{A}} \eta^t_{a\to a'} \geq 0 \qquad\qquad \forall a \in \mathcal{A},\, \forall t \in \mathcal{T}\,.$$

Given an allocation $\boldsymbol{\pi}$, there are many admissible exchanges leading to the same outcome after we apply all of the prescribed exchanges. We illustrate this phenomenon in a short Example 5.2.

> **Example 5.2.** Consider five resources of a single resource type, two agents a and b, and an initial allocation $\boldsymbol{\pi} = (\pi^1_a, \pi^1_b) = (2, 3)$. Let $\boldsymbol{\eta} = (\eta_{a\to b}, \eta_{b\to a}) = (1, 2)$ be an admissible exchange. Applying $\boldsymbol{\eta}$ to $\boldsymbol{\pi}$, we obtain an allocation $\boldsymbol{\pi}' = (3, 2)$. We also obtain the same allocation $\boldsymbol{\pi}'$ by applying a different exchange $\boldsymbol{\eta}' = (2, 3)$ to $\boldsymbol{\pi}$.

In Example 5.2 we observe that what really matters for the result of applying an exchange is the difference in the agents' bundles. In other words, we only care about an overall impact of an exchange instead of particular exchanges that led to the final allocation. In the following definition, we formally express this observation. We introduce a *change* (as opposed to an exchange discussed so far) and additionally (analogously to Definition 5.3) we specify a condition that ensures that a particular change is applicable to a given allocation. To avoid confusion, we emphasize that from now on we will only be speaking about changes because they are more appealing to work with for our purpose.

Definition 5.4. Consider a set $\mathcal{A}$ of agents and a set of resources formed by a set $\mathcal{T}$ of resource types with each resource type $t \in \mathcal{T}$ having $\#(t)$ copies. A *change* is a vector $\boldsymbol{\Delta} := \left(\Delta^t_a\right)_{t\in\mathcal{T},a\in\mathcal{A}} \in \mathbb{Z}^{nm}$ such that, for every resource type $t \in \mathcal{T}$, $\sum_{a\in\mathcal{A}} \Delta^t_a = 0$. Given an allocation $\boldsymbol{\pi}$, the change $\boldsymbol{\Delta}$ is *admissible* if $0 \leq \pi^t_a + \Delta^t_a \leq \#(t)$.

The notion of a change is a convenient tool to define an efficiency concept we mostly focus on; namely, Pareto-efficiency. This concept intuitively means that if there is a possibility to change an allocation in a way that no agent is worse off and at least one is better off, then such an allocation should be modified because of being inefficient.

Definition 5.5. An allocation $\boldsymbol{\pi}$ is *Pareto-dominated* if there is an admissible change $\boldsymbol{\Delta}$ leading to an allocation $\tilde{\boldsymbol{\pi}} = \boldsymbol{\pi} + \boldsymbol{\Delta}$ such that:

$$\forall a \in \mathcal{A}\colon \quad \sum_{t \in \mathcal{T}} u_a(t)\tilde{\pi}_a^t \geq \sum_{t \in \mathcal{T}} u_a(t)\pi_a^t \quad \text{and} \tag{5.1}$$

$$\exists a \in \mathcal{A}\colon \quad \sum_{t \in \mathcal{T}} u_a(t)\tilde{\pi}_a^t > \sum_{t \in \mathcal{T}} u_a(t)\pi_a^t\,. \tag{5.2}$$

In fact, Pareto-dominated allocations are rather undesirable, since they can always be changed to allocations that are more "satisfying" for the agents. Thus, to achieve allocations that do not underutilize resources one should consider allocations that are not Pareto-dominated (in fact, in the literature both weaker and stronger notions of efficiency have been studied).

Definition 5.6. An allocation $\boldsymbol{\pi}$ is *Pareto-efficient* if it is not Pareto-dominated.

5.2.2 *N*-Fold Integer Programming

Recall the ILP problem as defined in Section 2.6. We pay special attention to some restrictions of ILP. Most importantly, we restrict the structure of the constraint matrix A. Let N be a positive integer. Then, N-Fold Integer Programming (N-Fold IP) is a restricted variant of ILP.[1] To define the N-Fold IP problem, we first introduce bimatrices and N-fold products in the following two definitions.

Definition 5.7. Let $T \in \mathbb{Z}^{r\times t}$ and $D \in \mathbb{Z}^{s\times t}$ be two matrices. A bimatrix $\binom{T}{D}$ is an $(r+s)\times t$ matrix containing matrix T as the top r rows and matrix D as the last s rows.

Definition 5.8. For some positive integers r, t, and s, let $T \in \mathbb{Z}^{r\times t}$ and $D \in \mathbb{Z}^{s\times t}$ be some matrices. An *N-fold product* $E^{(N)}$ of a bimatrix $E = \binom{T}{D}$ is the following matrix of size $(r + N\cdot s)\times(N\cdot t)$:

$$E^{(N)} = \tbinom{T}{D}^{(N)} = \begin{pmatrix} T & T & \cdots & T \\ D & 0 & \cdots & 0 \\ 0 & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D \end{pmatrix} \tag{5.3}$$

[1] Since the name N-fold IP stands for N-fold Integer Programming, the above sentence is not precise. This is because the goal function in N-Fold IPs does not need to be linear, while, by definition, it is linear in the ILP problem. We could have written N-Fold ILP but we decided to keep the traditional name.

Using an N-fold product matrix as a constraint matrix in the ILP problem, we obtain the N-Fold Integer Programming problem.

> N-Fold Integer Programming (N-Fold IP)
> **Input:** An integral constraint matrix $A^{(N)} \in \mathbb{Z}^{(r+N\cdot s)\times(N\cdot t)}$ being an N fold product of a bimatrix $A = \binom{T}{D}$, where $T \in \mathbb{Z}^{r\times t}$ and $D \in \mathbb{Z}^{s\times t}$, a right-hand side vector $\mathbf{b} \in \mathbb{Z}^{r+N\cdot t}$, two boundary vectors $\mathbf{l} \in \mathbb{Z}^{N\cdot t}$, $\mathbf{u} \in \mathbb{Z}^{N\cdot t}$, and a vector $\mathbf{w} \in \mathbb{Z}^{N\cdot t}$ representing a linear function.
> **Task:** Find a vector $\mathbf{x} \in \mathbb{Z}^{N\cdot t}$ of integers that minimizes $\mathbf{w}^\intercal\mathbf{x}$ subject to $A^{(N)}\mathbf{x} = \mathbf{b}$ and $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$.

A special case of N-Fold IP is the so-called Combinatorial N-Fold IP problem. Therein, one further restricts the input in such a way that $D = \mathbf{1}^\intercal$, that is, the matrix D has only one row which is the all-ones vector [KKM20a].

The first algorithms to solve N-Fold IP relied on the concept of the Graver basis [HKW10, HOR13]. In what follows, we discuss only the notions directly needed in this chapter; for a more detailed discussion of the topic see the corresponding monographs [LHK13, Onn10].

Definition 5.9. Let $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$ be two n-dimensional integer vectors. We write $\mathbf{y} \sqsubseteq \mathbf{x}$ if, for each $i \in [n]$, $x_i y_i \geq 0$ and $|y_i| \leq |x_i|$.

Observe that $\sqsubseteq$ imposes a partial order on n-dimensional vectors.

Definition 5.10. For an integer matrix $A \in \mathbb{Z}^{m\times n}$, its *Graver basis* $\mathcal{G}(A)$ is the set of $\sqsubseteq$-minimal non-zero elements in the set $\{\mathbf{z} \in \mathbb{Z}^n \mid A\mathbf{z} = \mathbf{0}\}$.

Below, we show an example demonstrating Definition 5.10.

> **Example 5.3.** Let A be an all-ones matrix of dimension 1×4. Consider some vector $\mathbf{x} \coloneqq (0, 0, -1, 1)$. Then, naturally, $A\mathbf{x} = \mathbf{0}$. The product of matrix A and a vector $\mathbf{y} \coloneqq (1, -1, -1, 1)$ is also $\mathbf{0}$. However, $\mathbf{x} \sqsubseteq \mathbf{y}$ and $\mathbf{y} \not\sqsubseteq \mathbf{x}$, thus $\mathbf{y}$ is not a $\sqsubseteq$-minimal element. Similarly, scaling $\mathbf{x}$ by any constant $c \in \mathbb{Z}$ such that $|c| > 1$ yields a vector $\mathbf{z}$ such that $A\mathbf{z} = \mathbf{0}$, $\mathbf{x} \sqsubseteq \mathbf{z}$, and $\mathbf{z} \not\sqsubseteq \mathbf{x}$. Finally, a vector $\mathbf{q} \coloneqq (-1, 1, 0, 0)$, for which it also holds that $A\mathbf{q} = \mathbf{0}$, is incomparable (according to $\sqsubseteq$) to $\mathbf{x}$, and vice versa.

The Graver basis constitutes the so-called *test set* [Gra75] for ILPs as formally presented in Proposition 5.1. The name comes from the fact that, for some

arbitrary feasible solution of an integer linear program, the solution is optimal if there is no vector in the Graver basis whose addition leads to another feasible solution that yields a smaller value of the goal function.

Proposition 5.1 ([Gra75])**.** *Consider a feasible solution* $\hat{\mathbf{x}}$ *to some instance of* ILP *with a vector* $\mathbf{w}$ *representing the goal function. Then, either of the following is true:*

- $\hat{\mathbf{x}}$ *is an optimal solution or*
- *there exists a vector* $\mathbf{g} \in \mathcal{G}(A)$ *such that* $\hat{\mathbf{x}} + \mathbf{g}$ *is feasible for the given* ILP *and* $\mathbf{w}^\intercal \hat{\mathbf{x}} > \mathbf{w}^\intercal(\hat{\mathbf{x}} + \mathbf{g})$.

In the subsequent results, we use a specific structure of solutions to N-FOLD IP in order to upper-bound the number of the elements of the Graver basis.

Definition 5.11. Consider an instance of N-FOLD IP with a constraint matrix $E^{(N)}$ being an N-fold product of some matrices $D \in \mathbb{Z}^{r \times t}$ and $A \in \mathbb{Z}^{s \times t}$. Every potential solution vector $\mathbf{x}$ can be expressed as $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ where each $\mathbf{x}_i \in \mathbb{Z}^t$, $i \in [N]$, is called a *brick*.

The key property of the N-fold product is that the number of *nonzero bricks*—that is, bricks that do not consist solely of zeros—in its Graver basis is upper-bounded by a constant that does not depend on N. Furthermore, this constant is an upper bound of the ℓ_1-norm of every vector in the Graver basis. We state these results (proven and stated in multiple forms [HOR13, HS07, Onn10]) in the following proposition.

Proposition 5.2. *For some natural number* N*, let* $E^{(N)}$ *be an* N*-fold product of a bimatrix* E*. The number of nonzero bricks in every* $\mathbf{g} \in \mathcal{G}\left(E^{(N)}\right)$ *is tightly upper-bounded by a constant* $g(E)$ *called the* Graver complexity *of* E*. Furthermore, it holds that* $\|\mathbf{g}\|_1 \leq g(E)$.

Our final step towards bounding the Graver complexity of the bimatrix E is to show how large $g(E)$ can be depending on E. We proceed using the bound on $g(E)$ that has been proven for N-fold IP [EHK18]. Since we use this bound only for combinatorial N-fold IPs, we simplify it here. Recall that in combinatorial N-fold IPs, the matrix D on the diagonal of $E^{(N)}$ has only one row of ones. In this case, observe that the ℓ_1-norm of an element of the Graver basis of $\mathbf{1}^\intercal$ is 2. Indeed, as indicated in Example 5.3, its Graver basis consists of vectors having exactly two nonzero entries of which one is set to -1 and the

other to 1. Using the just-established upper bound of the ℓ_1-norm of an element of the Graver basis of matrix $\mathbf{1}^\intercal$, we derive an upper-bound on the Graver complexity of the (whole) bimatrix $E = \left(\begin{smallmatrix} T \\ D \end{smallmatrix}\right) = \left(\begin{smallmatrix} T \\ \mathbf{1}^\intercal \end{smallmatrix}\right)$. To obtain this bound, shown in the following Proposition 5.3, we apply Lemmata 2 and 3 by Eisenbrand, Hunkenschröder, and Klein [EHK18] to our special case of interest. Therein, an upper bound of the ℓ_1-norm of an element of the Graver basis of the matrix D in bimatrix E is referred to as L_B. Since in our case this ℓ_1-norm upper bound (as we have shown above) is 2, we substitute "their" L_B with 2 and we arrive at the following proposition.

Proposition 5.3 ([EHK18]). *Let $T \in \mathbb{Z}^{r\times t}$ and $\Delta = \|T\|_\infty$. Then, $g(\left(\begin{smallmatrix} T \\ \mathbf{1}^\intercal \end{smallmatrix}\right)) \leq 2(4r\Delta + 1)^r$.*

Intuitively, Proposition 5.1 upper-bounds the Graver complexity of a combinatorial N-fold integer program by a function solely of the number of constraints and the maximum absolute value of each entry in the constraint matrix—note, that the number of variables of the integer program is irrelevant. The Graver complexity also upper-bounds the ℓ_1-norm of elements of the Graver basis of the constraint matrix (see Proposition 5.2). Thus, if the value of the upper bound is "sufficiently small," then one can enumerate the whole Graver basis and use its elements to iteratively improve a solution to an N-fold integer program (see Proposition 5.1) with an unbounded number of variables, possibly much larger than the number of constraints.

5.2.3 Presburger Arithmetic

Presburger arithmetic is a helpful tool to represent, transform and combine multiple integer linear programs. Intuitively, Presburger arithmetic [Pre29] is a logical language for defining arithmetical properties of integers. Presburger arithmetic is a first-order theory over the domain of natural numbers with atoms being linear equations. The formulas of Presburger arithmetic are then built using standard logical symbols.

To start with, let us consider a simple linear equation where we ask whether there is an integer vector $\mathbf{x}$ such that for a given integer vector $\mathbf{a}$ and an integer b equation $\mathbf{a}^\intercal\mathbf{x} = b$ holds. We can model the aforementioned problem of checking whether a linear equation is solvable with the following Presburger arithmetic sentence:

$$\varphi = (\exists \mathbf{x})(\mathbf{a}^\intercal\mathbf{x} = b).$$

Naturally, asking whether φ is true is equivalent to asking whether the above-mentioned linear equation has a solution.

Now, let us recall the Integer Linear Program Feasibility problem (ILPF) in which we ask whether for some integer constraint matrix A, an integer right-hand side vector $\mathbf{b}$, and boundary vectors $\mathbf{l}$, $\mathbf{u}$ there is a vector $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$ and $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$. Let r be the number of rows of matrix A, and, for each $i \in [r]$, let $\mathbf{a}_i$ be the i-th row of A. Then, answering such an instance of ILPF can be alternatively presented as checking whether the following sentence in Presburger arithmetic is true:

$$(\exists \mathbf{x}) \left(\bigwedge_{i \in [r]} \mathbf{a}_i \mathbf{x} = b_{\mathbf{a}} \right). \tag{ILP:PA}$$

Naturally, we can build more complicated Presburger arithmetic sentences. We can use the universal quantifier, build a chain of quantifiers, and negate atoms. Even though we indeed use more complicated sentences in this thesis, we will always finally arrive at sentences in the form of (ILP:PA).

5.3 Seeking Envy-Free Pareto-Efficient Allocations

The following basic problem [BBN16] is the starting point for our considerations in this section.

EEF–Allocation
Input: A set $\mathcal{A}$ of agents, a set $\mathcal{T}$ of resource types, agent utilities $u_a \colon \mathcal{T} \to \mathbb{Z}$ for every $a \in \mathcal{A}$, and resource multiplicities $\#(t) \in \mathbb{N}$ for $t \in \mathcal{T}$.
Question: Is there an envy-free Pareto-efficient allocation?

EEF–Allocation is known to be Σ_2^p-complete and remains NP-hard for several restricted cases [BL08]. Our goal in this section is to study EEF–Allocation from a parameterized viewpoint focusing on the parameterization by the number of agents plus the maximum utility (specifically, its absolute value). We show that for this selection of parameters EEF–Allocation is fixed-parameter tractable. We mention that our results are constructive, thus we not only can give positive algorithmic results for the decision version of EEF–Allocation but also for the corresponding search or optimization problems.

Before we briefly discuss our choice of the parameterization, let us settle some notation used in the remainder of this chapter and link it to our toy example

in Figure 5.2. We refer to the number of agents, that is three in the example, as n; formally, $n \coloneqq |\mathcal{A}|$. We denote the number of resource types, that is five in the example, by h; formally, $h \coloneqq |\mathcal{T}|$. By $u_{\max}$ we mean the maximum absolute value of the utilities reported by the agents, that is four in our example; formally, $u_{\max} \coloneqq \max_{a \in \mathcal{A}} \max_{t \in \mathcal{T}} |u_a(t)|$.

Note that for n agents and the maximum (absolute) utility $u_{\max}$, the number $|\mathcal{T}|$ of possible resource types is at most $u_{\max}^n$. Thus, our parameterization is very "strong" in the sense that it severely restricts the input instances. However, this strong parameterization allows our algorithm to be fixed-parameter tractable in the so-called high-multiplicity regime, where the multiplicities of the resources are given in binary in the input. Our fixed-parameter tractability result, formally stated in the following Theorem 5.4, also answers two main open questions of Bliem, Bredereck, and Niedermeier [BBN16] (see the discussion in the last paragraph of Section 5.1).

Theorem 5.4. EEF–Allocation *with n agents and the maximum utility value $u_{\max}$ is fixed-parameter tractable when parameterized by $n + u_{\max}$.*

Before we prove Theorem 5.4 step-by-step in the subsequent sections, we describe its high-level idea, which intuitively illustrates our approach, together with the organization of the proof.

In Section 5.3.1, we begin with showing that if an allocation $\boldsymbol{\pi}$ is not Pareto-efficient (i.e., $\boldsymbol{\pi}$ is dominated by some allocation $\boldsymbol{\pi}'$), then it is dominated by an allocation $\hat{\boldsymbol{\pi}}$ which does not differ from $\boldsymbol{\pi}$ much (see Lemma 5.5). In other words, there exists a constant $c_{\max} = c_{\max}(n, u_{\max})$ such that if $\boldsymbol{\pi}$ is not dominated by any allocation $\boldsymbol{\pi}'$ with $\|\boldsymbol{\pi} - \boldsymbol{\pi}'\|_1 \leq c_{\max}$, then $\boldsymbol{\pi}$ is Pareto-efficient. Using this insight, in Section 5.3.2, we define a set of "small allocation-improving changes" $\mathcal{D}$, in a way that an allocation $\boldsymbol{\pi}$ is dominated if and only if there exists an change $\boldsymbol{\Delta} \in \mathcal{D}$ such that $\boldsymbol{\pi} + \boldsymbol{\Delta}$ is an allocation. Consequently, an allocation $\boldsymbol{\pi}$ is a solution for a given instance of EEF–Allocation if it is envy-free and, for all $\boldsymbol{\Delta} \in \mathcal{D}$, $\boldsymbol{\pi} + \boldsymbol{\Delta}$ is not an allocation. We can express an allocation as a relatively simple integer linear program in which both the number of variables and the number of constraints is upper-bounded by a function of the parameter $n + u_{\max}$. This in turn makes it possible to "negate" such a model, which we show in Section 5.3.3, while keeping the dimension upper-bounded in terms of a function the parameter (by introducing large coefficients). Finally, we collect the aforementioned negation integer linear programs for all $\boldsymbol{\Delta} \in \mathcal{D}$ and these (applied to $\boldsymbol{\pi} + \boldsymbol{\Delta}$) together with the integer linear program for envy-free allocation $\boldsymbol{\pi}$ yield the model for EEF–Allocation. The resulting model's

dimensions are upper-bounded by a function solely of the parameter, eventually yielding fixed-parameter tractability exploiting ILP-related results going back to that of Lenstra [Len83] from Proposition 2.1.

5.3.1 Locality of Dominance Test

We show that if an allocation of resources to agents is not Pareto-efficient, then it is dominated by an allocation that is "close" to it. To this end, we show the following.

Lemma 5.5. *Let $\mathcal{A}$ be a set of n agents, let $\mathcal{T}$ be a set of resource types, and let $\boldsymbol{\pi}$ be an allocation. If $\boldsymbol{\pi}$ is dominated with respect to agent utilities $u_a \colon \mathcal{T} \to \mathbb{Z}$, where $u_{\max}$ is the maximum absolute utility value, then there exists a change $\boldsymbol{\Delta}$ such that*

1. *change $\boldsymbol{\Delta}$ is admissible for $\boldsymbol{\pi}$,*
2. *the allocation $\boldsymbol{\pi} + \boldsymbol{\Delta}$ dominates $\boldsymbol{\pi}$, and*
3. *$\|\boldsymbol{\Delta}\|_1 \leq 2 \cdot (4n \cdot u_{\max} + 1)^n$.*

Proof Idea Before we give a formal proof of Lemma 5.5, we describe the idea behind it. First, we give an integer linear program which can be used to decide whether there exists an allocation dominating the given allocation $\boldsymbol{\pi}$. That is, we show that the following problem is fixed-parameter tractable with respect to the combined parameter $n + u_{\max}$.

> Allocation Dominance
> **Input:** A set of agents $\mathcal{A}$, a set of h resource types $\mathcal{T}$, agent utilities $u_a \colon \mathcal{T} \to \mathbb{Z}$ for every $a \in \mathcal{A}$, resource multiplicities $\#(t) \in \mathbb{N}$ for $t \in \mathcal{T}$, and an allocation $\boldsymbol{\pi} \in \mathbb{N}^{nh}$.
> **Question:** Is there an allocation that dominates $\boldsymbol{\pi}$?

Then, we show that one can modify the aforementioned ILP formulation (in a way similar to Knop, Koutecký, and Mnich [KKM20a]) so that the modified instance is in fact an instance of Combinatorial N-Fold IP. Finally, we conclude that if the given allocation $\boldsymbol{\pi}$ is dominated, then there exists a vector $\mathbf{g}$ in the Graver basis of the constraint matrix of the modified model which is witnessing this fact. Since we design our model such that vector $\mathbf{g}$ is in fact a change that, after applied, leads to an allocation dominating $\boldsymbol{\pi}$, the upper bound of $\|\boldsymbol{\Delta}\|_1$ from Lemma 5.5 follows from Proposition 5.3.

Proof of Lemma 5.5. We start with providing the following natural formulation of the Allocation Dominance problem. In order to find a dominating allocation $\mathbf{x}$, for every agent $a \in \mathcal{A}$ and every resource type $t \in \mathcal{T}$, we introduce an integral variable x_a^t that indicates how many resources of type t allocation $\mathbf{x}$ assigns to agent a.

$$\sum_{t \in \mathcal{T}} u_a(t) \cdot x_a^t \geq \sum_{t \in \mathcal{T}} u_a(t) \cdot \boldsymbol{\pi}_a^t \qquad \forall a \in \mathcal{A}, \tag{5.4}$$

$$\sum_{a \in \mathcal{A}} x_a^t \leq \#(t) \qquad \forall t \in \mathcal{T}, \tag{5.5}$$

$$0 \leq x_a^t \leq \#(t) \qquad \forall a \in \mathcal{A} \forall t \in \mathcal{T}. \tag{5.6}$$

Constraints (5.4) ensure that each agent a valuates the bundle that $\mathbf{x}$ assigns to a at least as good as the one it gets from $\boldsymbol{\pi}$. Constraints (5.5) and (5.6) prevent allocation $\mathbf{x}$ from assigning more resources than available. We add the following goal function

$$\max \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} \left(u_a(t) \cdot x_a^t \right) - \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} u_a(t) \cdot \boldsymbol{\pi}_a^t \,. \tag{5.7}$$

Note that the value of the goal function is 0 if we set $\mathbf{x} = \boldsymbol{\pi}$; hence, 0 constitutes a lower bound on the value of the goal function. However, if there is a solution $\mathbf{x}$ for which the value of the goal function is positive, then $\mathbf{x}$ dominates $\boldsymbol{\pi}$. [2]

Now, let us present the integer linear program defined by Inequalities (5.4), (5.5), and (5.6) using matrices. In the following matrix representation, we order the sought variables according to the resource types they correspond to; that is, we consider vector $\mathbf{x} \coloneqq (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^h)$ where each vector $\mathbf{x}^t \coloneqq (\mathbf{x}_{a_1}^t, \mathbf{x}_{a_2}^t, \ldots, \mathbf{x}_{a_n}^t)$. Representing the right-hand side vectors of Inequalities (5.4) and (5.5) as the appropriate vector $\mathbf{b}$, the upper bounds from Inequality (5.6) as the appropriate boundary vector $\mathbf{u}$ and, for each $t \in \mathcal{T}$, building the appropriate matrix $U_i \in \mathbb{Z}^{n \times n}$ expressing the respective conditions from Inequality (5.4), we arrive at the

[2] We note that at this point one may already derive an upper bound on $\|\mathbf{x}\|_\infty$ as a consequence of a result due to Papadimitriou [Pap81, Theorem]; namely, $\|\mathbf{x}\|_\infty \leq nh \cdot ((n+h) \cdot t_{\max})^{2(n+h)+1}$, where $t_{\max} \coloneqq \max_{t \in \mathcal{T}} \#(i)$. Note that this upper bound is worse than the one we claim. In particular, our bound is independent of the number of resources.

following matrix form of the above integer linear program:

$$\begin{pmatrix} U_1 & U_2 & \cdots & U_h \\ \mathbf{1}^\intercal & 0 & \cdots & 0 \\ 0 & \mathbf{1}^\intercal & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{1}^\intercal \end{pmatrix} \cdot \mathbf{x} \leq \mathbf{b}\,,$$
$$\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}\,.$$

Observe that such a formulation resembles N-Fold IP; in fact, it is called a combinatorial pre-N-fold IP. Our goal now is to find a small matrix U which we can use to "embed" all of the matrices U_i. Observe that the matrices U_i have diagonal form and that $\|U_i\|_\infty \leq u_{\max}$ holds. Thus, let us consider the following matrix $U \in \mathbb{Z}^{n \times (n \cdot (2u_{\max}+1))}$.

$$U \coloneqq \begin{pmatrix} -u_{\max} & \cdots & u_{\max} & 0 & \cdots & 0 & \cdots & 0 & & 0 \\ 0 & \cdots & 0 & -u_{\max} & \cdots & u_{\max} & \cdots & 0 & & 0 \\ \vdots & & & & & & \ddots & & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & -u_{\max} & \cdots & u_{\max} \end{pmatrix}.$$

We show a way to use the above-defined matrix U to substitute each matrix U_i so that we obtain a Combinatorial N-Fold IP equivalent to the original ILP that we started with in this section. Furthermore, the new Combinatorial N-Fold IP will be "small enough" to use Proposition 5.3 and finally prove Lemma 5.5.

The matrix U is designed in such a way that, for every $t \in \mathcal{T}$ and the corresponding matrix U_i, there is a set of columns of U that, if taken out and assembled in order to construct a new matrix, form U_i. It is enough to go through U_i column by column and select the same-looking columns from each block of matrix U. We mimic this kind of selection in two steps. First, we replace all matrices U_i with a copy of U and, as a consequence, we add additional variables by substituting each variable $x^t_{a_j}$, where $a_j \in \mathcal{A}$ and $t \in \mathcal{T}$, with a

vector $\hat{\mathbf{x}}^t_{a_j}$ as depicted below:

$$\begin{pmatrix} U_1 & U_2 & \cdots & U_h \\ \mathbf{1}^\intercal & 0 & \cdots & 0 \\ 0 & \mathbf{1}^\intercal & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{1}^\intercal \end{pmatrix} \begin{pmatrix} x^1_{a_1} \\ x^1_{a_2} \\ \vdots \\ x^1_{a_n} \\ x^2_{a_1} \\ \vdots \end{pmatrix} \rightsquigarrow \begin{pmatrix} U & U & \cdots & U \\ \mathbf{1}^\intercal & 0 & \cdots & 0 \\ 0 & \mathbf{1}^\intercal & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{1}^\intercal \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}^1_{a_1} \\ \hat{\mathbf{x}}^1_{a_2} \\ \vdots \\ \hat{\mathbf{x}}^1_{a_n} \\ \hat{\mathbf{x}}^2_{a_1} \\ \vdots \end{pmatrix},$$

$$\text{where} \quad \hat{\mathbf{x}}^t_{a_j} \coloneqq \left(x^t_{a_j,-u_{\max}}, x^t_{a_j,-u_{\max}+1}, \ldots, x^t_{a_j,u_{\max}}\right).$$

Second, using the boundary constraints we "neglect" some (actually a vast number) of the extra variables, that is, we set both their lower bound and upper bound to 0. Observe that, for $a_j \in \mathcal{A}$ and $t \in \mathcal{T}$, the variables in vector $\hat{\mathbf{x}}^t_{a_j}$ serve as selectors of such a column $\mathbf{c}$ from the respective copy of matrix U (the copy used to substitute U_t) that is identical to the j-th column in U_t. Selecting such a column $\mathbf{c}$ is equivalent to letting variable $x^t_{a_j,y}$ such that $y = u_{a_j}(t)$ mimic variable $x^t_{a_j}$ and setting all other variables from $\hat{\mathbf{x}}^t_{a_j}$ to 0. Thus, we introduce new constants $\chi^t_{a,u}$, where $\chi^t_{a,u} \coloneqq 1$ if $u_a(t) = u$ and $\chi^t_{a,u} \coloneqq 0$, otherwise. Now, the boundary vectors can be expressed by the following inequalities:

$$0 \leq x^t_{a,u} \leq \chi^i_{a,u} \cdot \#(i).$$

Performing the described substitutions leads to an ILP formulation that is very similar to the Combinatorial N-Fold IP. However, it still contains inequalities coming from Constraints (5.4) and (5.5). Thus, according to our definition of Combinatorial N-Fold IP, we need to further transform the model to obtain equalities from the aforementioned inequalities (note that the constrains involving boundary vectors are always given as inequalities). Thus, we add one slack variable for each Constraint (5.4) and one slack variable for each Constraint (5.5). Observe that to keep the N-Fold IP structure, adding one slack variable to each Constraint (5.4)—which is a single row in the copies of U—requires adding it to all copies of U. Thus, we need to augment each copy of the matrix U with an identity matrix of dimension $n \times n$. Consequently, we obtain an excessive number of slack variables (note that in each constraint coming from Constraint (5.4) we add h slack variables). Hence, we can use the excessive variables as slack variables for constraints coming

from Inequality (5.5). Eventually, applying all the transformations described above, we arrive at Combinatorial N-Fold IP with the following values of parameters from Proposition 5.3:

- $r \coloneqq n$,
- $t \coloneqq 2n \cdot (u_{\max} + 1) + n$,
- $\|T\|_\infty \coloneqq u_{\max}$, and
- $N \coloneqq h$.

By computing the upper bound according to the above values, we obtain the desired upper bound on the ℓ_1-norm of an element of the Graver basis because each vector of the Graver basis corresponds to an admissible change for $\boldsymbol{\pi}$. □

5.3.2 Modeling the Existence of Envy-Free Efficient Allocations

We use Lemma 5.5 from the previous section to express the existence of a solution to the EEF–Allocation problem. To do so we first represent the existence of such an allocation as a "complicated" sentence in the framework of Presburger arithmetic. Afterwards, we apply several transformations to bring the sentence to the form of Sentence (ILP:PA).

Potential Changes Leading to Domination

Basing on the bound given by Lemma 5.5, we define the set $\mathcal{D}$ of all possible changes that, if admissible for some allocation $\boldsymbol{\pi}$, would yield an allocation that dominates $\boldsymbol{\pi}$. To make our arguments simpler, we add a special agent $\diamond$ whose purpose is to collect all resources that are unassigned to the agents in $\mathcal{A}$. Let $A^\diamond$ denote the set $\mathcal{A} \cup \{\diamond\}$ and let $c_{\max} \coloneqq 2 \cdot (4(n+1) \cdot u_{\max} + 1)^{n+1}$ be the upper-bound from Lemma 5.5 (note that our set of agents is now augmented by $\diamond$). Then, the set $\mathcal{D}$ can be defined as follows:

$$\mathcal{D} \coloneqq \left\{ \boldsymbol{\Delta} \in \mathbb{Z}^{h\cdot(n+1)} \colon \|\boldsymbol{\Delta}\|_1 \leq c_{\max} \wedge \left(\sum_{a\in\mathcal{A}} \sum_{t\in\mathcal{T}} u_a(t) \cdot \Delta_a^t \geq 1 \right) \wedge \right.$$
$$\left. \wedge \bigwedge_{a\in\mathcal{A}} \left(\sum_{t\in\mathcal{T}} u_a(t) \cdot \Delta_a^t \geq 0 \right) \wedge \bigwedge_{t\in\mathcal{T}} \left(\sum_{a\in A^\diamond} \Delta_a^t = 0 \right) \right\}. \tag{5.8}$$

Thanks to agent $\diamond$, we could require that, for each resource type $t \in \mathcal{T}$, a change $\boldsymbol{\Delta}$ fulfills $\sum_{a\in A^\diamond} \Delta_a^t = 0$. Naturally, $|\mathcal{D}| \leq (2c_{\max} + 1)^{h(n+1)}$. Furthermore, it is not hard to generate $\mathcal{D}$, since it is straightforward to verify the given conditions for all vectors with $\|\boldsymbol{\Delta}\|_1 \leq c_{\max}$.

Existence of Envy-Free Efficient Allocation in Presburger Arithmetic

Having the set $\mathcal{D}$, we can say that some envy-free allocation $\boldsymbol{\pi}$ is Pareto-efficient if there is no change in $\mathcal{D}$ that is admissible for $\boldsymbol{\pi}$. Our goal in this section is to formally express this observation in Presburger arithmetic, which will help us to show that we can construct an integer linear program modeling the existence of envy-free efficient allocations. To this end, we use three logical predicates $\text{allocation}(\mathbf{x})$, $\text{allocation}^*(\mathbf{x})$, and $\text{EF}(\mathbf{x})$. Intuitively, the first two are true if and only if $\mathbf{x}$ is an allocation and the last one becomes true if and only if $\mathbf{x}$ is an envy-free allocation. Using these operators, we express the desired observation as the following logical sentence φ:

$$\varphi \coloneqq (\exists \mathbf{x}) \left(\text{allocation}(\mathbf{x}) \wedge \text{EF}(\mathbf{x}) \wedge (\forall \boldsymbol{\Delta} \in \mathcal{D}) \left(\neg \, \text{allocation}^*(\mathbf{x} + \boldsymbol{\Delta}) \right) \right). \tag{5.9}$$

Notice that this is a natural description of the EEF–Allocation problem in "high-level" Presburger arithmetic.

The rest of this section is devoted to formally describing the predicates used in the above definition of φ.

- $\text{allocation}(\mathbf{x})$: We verify that $\mathbf{x}$ is a non-negative vector and that every resource is allocated to at least one agent (including the artificial agent $\diamond$). Thus, we have

$$\text{allocation}(\mathbf{x}) \coloneqq \bigwedge_{t \in \mathcal{T}} \left(\sum_{a \in A^\diamond} x_a^t = \#(t) \right) \wedge \bigwedge_{a \in A^\diamond, t \in \mathcal{T}} \left(x_a^t \geq 0 \right).$$

- $\text{EF}(\mathbf{x})$: We check the natural valuation condition for every pair of agents in $\mathcal{A}$ (note that $\diamond \notin \mathcal{A}$); hence, the following condition:

$$\text{EF}(\mathbf{x}) \coloneqq \bigwedge_{a, a' \in \mathcal{A}} \left(\sum_{t \in \mathcal{T}} u_a(t) \cdot x_a^t \geq \sum_{t \in \mathcal{T}} u_a(t) \cdot x_{a'}^t \right).$$

- $\text{allocation}^*(\mathbf{x} + \boldsymbol{\Delta})$: We check whether $\mathbf{z} \coloneqq \mathbf{x} + \boldsymbol{\Delta}$ is an allocation, which is equivalent to checking whether $\boldsymbol{\Delta}$ is admissible to allocation $\mathbf{x}$ (see Definition 5.3). Note that because we added agent $\diamond$, we know that $\mathbf{x}$ distributes all resources and that $\boldsymbol{\Delta}$ only redistributes these resources. Consequently, applying change $\boldsymbol{\Delta}$ leads to an agent having a negative number of some resource of type t if and only if some other agent has more than $\#(t)$ copies

of resource type t. Thus, we check only one of these conditions in the following definition of the predicate:

$$\text{allocation}^*(\mathbf{z}) \coloneqq \bigwedge_{a \in \mathcal{A}^\diamond, t \in \mathcal{T}} \left(z_a^t \geq 0\right).$$

Due to the definition of $\mathcal{D}$ in Formula (5.9), if $\mathbf{z}$ is an allocation, then indeed allocation $\mathbf{z}$ dominates $\mathbf{x}$.

5.3.3 Obtaining and Solving ILP Formulation

The goal of this part of the proof is to express formula φ from Section 5.3.2 as an integer linear program that has a number of variables that is upper-bounded by a function of parameter $n + u_{\max}$. We do this in two steps. The first one is to ensure that we can express φ with an equivalent formula that is a conjunction of atoms (as we will show, φ contains disjunctions of atoms) that allows for obtaining an ILP. The second, final step of the proof is to show that we can apply a result of Lenstra [Len83] (presented in Proposition 2.1) to show EEF–Allocation in the time yielding the requested FPT result from Theorem 5.4.

Obtaining ILP Formulation by Negating the Allocation Predicate

Since $\text{allocation}^*(\mathbf{x}+\mathbf{\Delta})$ is a conjunction of atoms, we observe that if we negate it, then we arrive at a disjunction of atoms. This is not desirable for our purposes of modeling our problem as an ILP instance. However, it is possible to transform the disjunctions into conjunctions for a fixed (given) $\mathbf{\Delta}$ using large coefficients.

Note that the negation of $\text{allocation}^*(x_a^t)$ is true if and only if there exists an agent a and a resource type t such that either $x_a^t - \Delta_a^t$ is negative or $x_a^t - \Delta_a^t > \#(t)$. We can model this expression as follows. Let $\#_{\max} \coloneqq \max_{t \in \mathcal{T}} \#(t)$ and fix $a \in \mathcal{A}$ and $t \in \mathcal{T}$. We add a binary variable y_a^t that takes value 1 if $x_a^t + \Delta_a^t < 0$. We achieve this via the following constraint:

$$x_a^t + \Delta_a^t \leq -1 + (\#_{\max} + c_{\max} + 1)(1 - y_a^t). \tag{5.10}$$

The inequality above is universally fulfilled when $y_a^t = 0$, since we have $0 \leq x_a^t \leq m_{\max}$ and $\left|\Delta_a^t\right| \leq c_{\max}$. However, y_a^t can be set to 1 without violating Constraint (5.10) only if $x_a^t + \Delta_a^t < 0$; thus showing that change $\mathbf{\Delta}$ does not lead to an allocation.

Observe that if we put together Constraint (5.10) applied to all pairs $a \in \mathcal{A}$, $t \in \mathcal{T}$, then the condition

$$\sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} y_a^t \geq 1 \tag{5.11}$$

ensures that at least one variable y_a^t is set to 1. This, in turn, implies that $\mathbf{x} + \mathbf{\Delta}$ is not an allocation. For brevity, we encapsulate the collection of constraint resulting from applying Constraint (5.10) for all pairs $a \in \mathcal{A}, t \in \mathcal{T}$ together with inequality (5.11) in the predicate certificate$(\mathbf{x} + \mathbf{\Delta}, \mathbf{y})$. Naturally, certificate$(\mathbf{x} + \mathbf{\Delta}, \mathbf{y})$ is a conjunction of atoms and, if true, certifies that $\mathbf{x} + \mathbf{\Delta}$ is not an allocation.

Finally, we incorporate the certificate$(\mathbf{x} + \mathbf{\Delta}, \mathbf{y})$ predicate into the sentence φ, thus arriving at the following equivalent form of it:

$$\hat{\varphi} \coloneqq (\exists \mathbf{x}) \left(\text{allocation}(\mathbf{x}) \wedge \text{EF}(\mathbf{x}) \wedge (\forall \mathbf{\Delta} \in \mathcal{D}) \left(\exists \mathbf{y}^{\mathbf{\Delta}} \right) \left(\text{certificate}(\mathbf{x} + \mathbf{\Delta}, \mathbf{y}^{\mathbf{\Delta}}) \right) \right).$$

Using Lenstra's Algorithm

A characteristic feature of $\hat{\varphi}$ is that, for each $\mathbf{\Delta} \in \mathcal{D}$, the corresponding vector $\mathbf{y}^{\mathbf{\Delta}}$ is independent of other vectors $\mathbf{y}^{\mathbf{\Delta}'}$. Thus, defining a vector $\mathbf{y} \coloneqq \left(y^{\mathbf{\Delta}}\right)_{\mathbf{\Delta} \in \mathcal{D}}$ of certificates, we arrive at the following formula φ_{ILP} that is equivalent to $\hat{\varphi}$:

$$\varphi_{\text{ILP}} \coloneqq (\exists \mathbf{x}) (\exists \mathbf{y}) \left(\text{allocation}(\mathbf{x}) \wedge \text{EF}(\mathbf{x}) \wedge \bigwedge_{\mathbf{\Delta} \in \mathcal{D}} \text{certificate}(\mathbf{x} + \mathbf{\Delta}, \mathbf{y}^{\mathbf{\Delta}}) \right). \quad (5.12)$$

Observe that the existential sentence φ_{ILP} indeed describes an integer linear program over variable vectors $\mathbf{x}$ and $\mathbf{y}$. Invoking Lenstra's algorithm (Proposition 2.1) we can solve the integer linear program defined by (5.12) in FPT-time with respect to the number of the program's variables. Note that the number d of variables of φ_{ILP} is upper-bounded by

$$\begin{aligned} d &= (n+1) \cdot h + 2h(n+1) \cdot |\mathcal{D}| \leq (n+1) \cdot h + 2h(n+1) \cdot (2c_{\max} + 1)^{h(n+1)} \\ &= (n+1) \cdot h + 2h(n+1) \cdot (2 \cdot 2 \cdot (4n \cdot u_{\max} + 1)^n)^{h(n+1)}. \end{aligned} \quad (5.13)$$

Since $h \leq (2u_{\max} + 1)^n$, we obtain the fixed-parameter tractability of finding an envy-free and Pareto-efficient allocation with respect to the number n of agents plus the maximum absolute value $u_{\max}$ of utility—which concludes the proof of Theorem 5.4.

5.3.4 Relaxing Pareto-Efficiency

The algorithm described so far in Section 5.3 finds an envy-free Pareto-efficient allocation only if such an allocation exists. In case there is no such allocation, the algorithm only reveals this fact. However, in practice, in that case we would

rather be interested in finding an allocation that is (in some sense) "almost" Pareto-efficient. This is why we conclude this section by showing one approach of tweaking the algorithm to work in this way.

Recall that, in Section 5.3.3, in order to write the predicate certificate$(\cdot)$ we added numerous variables $\mathbf{y}$ (for a fixed $\boldsymbol{\Delta}$) to certify that at least one of the certificate conditions holds (i.e., at least one condition in the original system does not hold). To do so (for a fixed $\boldsymbol{\Delta}$) we added the constraint $\mathbf{1}^\intercal\mathbf{y} \geq 1$ (see Constraint (5.10)) requiring that there is always at least one certificate guaranteeing that the change $\boldsymbol{\Delta}$ does not lead to a proper allocation. Such a certificate, naturally, cannot be found if there is no Pareto-efficient envy-free allocation. However, instead of simply acknowledging that no solution exists, we show how to find an allocation that is "not far" from Pareto-efficiency.

To this end, we change condition $\mathbf{1}^\intercal\mathbf{y} \geq 1$ to $\mathbf{1}^\intercal\mathbf{y} \leq 1$ and take advantage from the possibility to maximize a linear goal function in the integer program φ_{ILP} from Section 5.3.3. Thus, if we maximize

$$\sum_{\boldsymbol{\Delta} \in \mathcal{D}} \mathbf{1}^\intercal \mathbf{y}^{\boldsymbol{\Delta}},$$

then we are seeking a solution that maximizes the number of valid certificates in our search space $\mathcal{D}$ of possible changes. Note that the value of the above expression is upper-bounded by $|\mathcal{D}|$ and is equal to it if there exists a Pareto-efficient envy-free allocation. If there is no Pareto-efficient envy-free allocation, then the modified ILP formulation no longer only reveal nonexistence, but it computes an envy-free allocation that has the fewest "small" changes leading to a dominating allocation.

Importantly, the modification presented in this section does not affect the requirement that the returned allocation has to be envy-free. We also do not require that the computed allocation is complete. Indeed, we did not add constraints on the artificial agent $\diamond$ in φ_{ILP} that would require that we cannot leave resources unallocated. So, in the worst case, it may happen that the returned allocation does not allocate any resource. This will be the case, for instance, for n agents and $n-1$ resources valued positively by every agent. Certainly, obtaining an empty, trivial allocation for the presented scenario is highly undesirable, yet due to the requirement of envy-freeness it is also unavoidable. The way to go to deal with such tricky cases is to relax envy-freeness, and how to do it within our framework is the topic of the next section.

5.4 Beyond Envy-Freeness

The idea behind the proof in Section 5.3 is quite universal. Indeed, it is possible to widely extend it to more general (broader) settings that consider different fairness and efficiency notions [Bre+19b, Theorem 2]. In this thesis, however, we restrict our considerations to different relaxations of envy-freeness; thus, we define the following Efficient $\mathcal{F}$-Allocation, where $\mathcal{F}$ is a placeholder for various "envy-freeness concepts."

> Efficient $\mathcal{F}$-Allocation [Bre+19b]
> **Input:** A set $\mathcal{A}$ of agents, a set $\mathcal{T}$ of resource types, agent utilities $u_a \colon \mathcal{T} \to \mathbb{Z}$ for every $a \in \mathcal{A}$, and resource multiplicities $\#(i) \in \mathbb{N}$ for $t \in \mathcal{T}$.
> **Question:** Is there an $\mathcal{F}$-free allocation which is Pareto-efficient?

By identifying important properties of $\mathcal{F}$, one can eventually obtain the following general results for the whole family of problems.

Theorem 5.6 (Bredereck et al. [Bre+19b])**.** Efficient $\mathcal{F}$-Allocation*, with n agents and the maximum absolute utility value $u_{\max}$, parameterized by $n + u_{\max}$ is fixed-parameter tractable if* $\mathcal{F}$-Allocation *admits an* ILP *formulation with $f(n + u_{\max})$ variables, where f is a computable function depending solely on the value of $n + u_{\max}$.*

We feature Theorem 5.6 by applying it to several different fairness concepts that have been considered in the literature so far. In this way, we show that the corresponding problems emerging from these concepts are also fixed-parameter tractable with respect to our parameter in our high-multiplicity regime.

In the subsequent sections, we study (and define) envy-freeness up to one good [Azi+19, BKV18, Car+16, Lip+04], envy-freeness up to any good [Azi+19, Car+16, PR18], and graph envy-freeness (studied also in Chapter 4). For all of them, we provide the corresponding ILP formulations compliant with Theorem 5.6. As a consequence, we obtain the following, general result.

Corollary 5.7. Efficient $\mathcal{F}$-Allocation*, with n agents and the maximum absolute utility value $u_{\max}$, parameterized by $n + u_{\max}$ is fixed-parameter tractable if $\mathcal{F}$ is (graph) envy-freeness, (graph) envy-freeness up to one good, or (graph) envy-freeness up to any good.*

5.4.1 Envy-Freeness up to Any Good

Allocations that are envy-free might not exist; for example, one cannot achieve envy-freeness when allocating a single resource among two agents that value it positively. To get around this natural limitation in practice, a slightly less demanding notion of envy-freeness, envy-freeness up to any good (EFX) [Azi+19, Car+16, PR18] has been introduced. Intuitively, under EFX (where X at the last position is meant to symbolically represent "any") agent a's envy towards agent a' is acceptable as long as the envy vanishes after pretending that a resource that *least* reduces the difference between a's own bundle utility and that of a' as seen by a does not exist. Clearly, each envy-free allocation is also an EFX allocation.

Definition 5.12 ([Azi+19])**.** An allocation π is called *envy-free up to any good* (EFX) if for each pair of (distinct) agents $a, a' \in \mathcal{A}$ with additive utility functions u_a and $u_{a'}$ it holds that

$$\forall r \in \pi(a') \wedge u_a(r) > 0\colon \quad u_a(\pi(a)) \geq u_a(\pi(a') \setminus \{r\}) \text{ and} \tag{5.14}$$

$$\forall r \in \pi(a) \wedge u_a(r) < 0\colon \quad u_a(\pi(a) \setminus \{r\}) \geq u_a(\pi(a')). \tag{5.15}$$

Note, that the conditions of EFX stated above use universal quantifiers. However, to check them it is enough to consider "the worst-case choice" of the resource to neglect. Fix some distinct agents a and a'. Naturally, in Condition (5.14) the worst-case choice is the resource with the minimum utility for agent a that is in the bundle of a'. In Condition (5.14), the worst-case choice is the resource with the maximum, but still negative, utility value for agent a in its own bundle.

In the remainder of this section we give an ILP model with at most $f(n + u_{\max})$ variables for checking whether an allocation is EFX. To this end, we fix some allocation π described by a vector $\mathbf{x} = \left(x_a^t\right)_{a \in \mathcal{A}, t \in \mathcal{T}}$ and then, for each pair of agents $a, a' \in \mathcal{A}$ and utility value $v \in [u_{\max}]$, we introduce binary variables representing the worst-case choices as follows:

1. If $a \neq a'$, then we introduce $x_{a,a'}^v$, which is 1 if and only if v is the minimum utility of a resource in the bundle of a' as seen by agent a;

2. if $a = a'$, then we introduce $x_{a,a}^{-v}$, which is 1 if and only if $-v$ (note the minus) is the maximum negative utility (closest to zero) of a resource in a's own bundle.

Our choice of variables reflects the two conditions in Definition 5.12. Observe that, for both conditions in Definition 5.12, the variable r bound by the universal quantifiers, which represents a resource, might not exist. For each pair of agents $a, a' \in \mathcal{A}$ we add special binary variables $x^0_{a,a'}$ that are 1 in such a degenerate case.

Before formally presenting the ILP formulation using the above-described variables, for each agent $a \in \mathcal{A}$ and utility value $v \in [u_{\max}]$, we define the following sets:

- $\mathcal{T}_a^{=v}$, which is the set of resource types with utility exactly v for a; formally, $\mathcal{T}_a^{=v} := \{t \in \mathcal{T} \mid u_a(t) = v\}$; and
- $\mathcal{T}_a^{\leq v}$, which is the set of resource types with utility at most v for a; formally, $\mathcal{T}_a^{\leq v} := \{t \in \mathcal{T} \mid u_a(t) \leq v\}$.

Using a "big" constant $M := \sum_{t \in \mathcal{T}} \#(t)$, we proceed with the following ILP model:

$$\sum_{t \in \mathcal{T}_a^{\leq v}} x^t_{a'} \leq M \cdot \left(\sum_{w=1}^{v} x^v_{a,a'} \right) \qquad \forall a, a' \in \mathcal{A}, a \neq a' \, \forall v \in [u_{\max}], \tag{5.16}$$

$$x^v_{a,a'} \leq \sum_{t \in \mathcal{T}_a^{=v}} x^t_{a'} \qquad \forall a, a' \in \mathcal{A}, a \neq a' \, \forall v \in [u_{\max}], \tag{5.17}$$

$$\sum_{v=0}^{u_{\max}} x^v_{a,a'} = 1 \qquad \forall a, a' \in \mathcal{A}, a \neq a', \tag{5.18}$$

$$\sum_{t \in \mathcal{T}_a^{\leq -v}} x^t_a \leq M \cdot \left(\sum_{w=-v}^{-1} x^{-v}_{a,a} \right) \qquad \forall a \in \mathcal{A} \, \forall v \in [u_{\max}], \tag{5.19}$$

$$x^{-v}_{a,a'} \leq \sum_{t \in \mathcal{T}_a^{=-v}} x^t_{a'} \qquad \forall a \in \mathcal{A}, a \neq a' \, \forall v \in [u_{\max}], \tag{5.20}$$

$$\sum_{v=1}^{u_{\max}} x^{-v}_{a,a} + x^0_{a,a} = 1 \qquad \forall a \in \mathcal{A}. \tag{5.21}$$

We show that Conditions (5.16), (5.17), and (5.18) indeed ensure the designed meaning of variables $x^v_{a,a'}$ and $x^0_{a,a'}$ for all utility values $v \in [u_{\max}]$ and distinct agents a and a'. Let us fix two distinct agents a and a'. Conditions (5.18) ensure that there is exactly one $v \in \{0\} \cup [u_{\max}]$ for which variable $x^v_{a,a'}$ equals 1.

Conditions (5.17) ensure that variable $x^v_{a,a'}$ can be 1 only if agent a' has (under allocation π) a resource to which a gives utility v. Finally, (for fixed a and a') to fulfill all inequalities in Condition (5.16), indeed the variable $x^v_{a,a'}$ for the smallest possible $v \in [u_{\max}]$ must be set to 1. Observe that $x^0_{a,a'}$ is 1 if and only if the left-hand sides of all conditions in (5.16) and the right-hand sides of all conditions in (5.17) are 0; this guarantees the intended meaning of $x^0_{a,a'}$ as an indicator that a' has no resources that are positively valued by a. The rest of the above-listed constraints analogously ensure the meaning of the variables of form $x^{-v}_{a,a}$. Note that Condition (5.19) uses the same technique as Condition (5.16), but by exchanging the summation borders it works in the opposite way.

Below we complement the above ILP model with two more condition sets which ensure that allocation π is EFX.

$$\sum_{t\in\mathcal{T}} x^t_a u_a(t) + M x^0_{a,a'} \geq \sum_{t\in\mathcal{T}} x^t_{a'} u_a(t) - \sum_{v=1}^{u_{\max}} v x^v_{a,a} \qquad \forall a, a' \in \mathcal{A}, a \neq a', \tag{5.22}$$

$$\sum_{t\in\mathcal{T}} x^t_a u_a(t) + M x^0_{a,a} \geq \sum_{t\in\mathcal{T}} x^t_{a'} u_a(t) - \sum_{v=1}^{u_{\max}} v x^{-v}_{a,a} \qquad \forall a, a' \in \mathcal{A}, a \neq a'. \tag{5.23}$$

Observe that Conditions (5.22) and (5.23) become trivially fulfilled if, respectively, $x^0_{a,a'}$ and $x^0_{a,a}$ are 1. Otherwise, we basically mimic the definition of EFX by neglecting one resource. Specifically, the resource that is "the worst-case choice" for being neglected. Hence, if neglecting this resource makes a pair of agents non-envious, then we know that this must be the case for all other resources.

5.4.2 Envy-Freeness up to One Good

It is an open question whether allocations that are envy-free up to any good (studied in the preceding section) always exist but it is conjectured that, in general, they might not exist [Car+16, CGM20]. Thus, to deal with such cases the notion of envy-freeness up to one good (EF1) [Azi+19, BKV18, Car+16, Lip+04] has been introduced. The intuition behind EF1 is similar to the one behind EFX. The difference is that this time we accept envy of a towards agent a' already if the resource that *most* reduces the difference between a's own bundle utility and that of a' as seen by a can be neglected to remove the envy of a. Naturally, each EFX allocation is also EF1. Importantly, allocations that are envy-free up to one good, unlike those that are envy-free or envy-free up to any good, always exist [Car+16]. Hence, they can serve as a fallback

in practical applications where, finally, "some" allocation has to be computed. In theoretical studies, the guarantee of their existence can be treated a "lower bound" on the fairness of allocations and thus might serve a yardstick for other fairness concepts.

Definition 5.13 ([Azi+19])**.** An allocation π is called *envy-free up to one good* (EF1) if for each pair of (distinct) agents $a, a' \in \mathcal{A}$ with additive utility functions, respectively, u_a and $u_{a'}$ either a does not envy a' or it holds that

$$\exists r \in \pi(a) \cup \pi(a') \colon \quad u_a(\pi(a) \setminus \{r\}) \geq u_a(\pi(a') \setminus \{r\}).$$

Observe that in contrast to EFX, where to fulfill its requirements one has to consider "the worst-case choice" of a resource to be neglected, in EF1 it is "the best-case choice" that is required by Definition 5.13. Based on this observation, we, again, give an ILP model with the number of variables upper-bounded by some (computable) function of $n + u_{\max}$ for checking whether an allocation is EF1 below.

Again, we fix some allocation π described by a vector $\mathbf{x} = \left(x_a^t\right)_{a \in \mathcal{A}, t \in \mathcal{T}}$. For each pair of agents $a, a' \in \mathcal{A}$ and utility value $v \in [u_{\max}]$, we introduce binary variables representing the candidate resources for the best-case choices (as opposed to the model of EFX where we were modeling the worst-case choices) as follows:

1. If $a \neq a'$, then we introduce $x_{a,a'}^v$, which is 1 if and only if v is the maximum utility of a resource in the bundle of a' as seen by agent a;
2. if $a = a'$, then we introduce $x_{a,a}^{-v}$, which is 1 if and only if $-v$ is the minimum utility of a resource in a's own bundle.

Again, we add two special variables to handle degenerate cases. First, for each agent $a \in \mathcal{A}$, we add variable $x_{a,a}^0$, which will be 1 if agent a has no resource with a negative utility in its bundle under π. Second, for each pair of distinct agents $a, a' \in \mathcal{A}$, variable $x_{a,a'}^0$ will be 1 if the bundle of agent a' does not contain a resource that has a positive value for agent a. Furthermore, for each pair of distinct agents $a, a' \in \mathcal{A}$ and each utility value $v \in [u_{\max}]$, we add one more variable $y_{a,a'}^v$, which will be 1 if utility value v is the utility of the best-case resource to be neglected for a and a'. Since for a pair of distinct agents $a, a' \in \mathcal{A}$ variables of the form $x_{a,a'}^v$ and $x_{a,a}^{-v}$ are computed independently of each other, the intuitive role of $y_{a,a'}^v$ is to aggregate their values and select one that finally constitutes "the best-case choice" of a resource to be neglected for agents a

and a'. We are now ready to lay out our ILP model for EF1 (that is similar to that of EFX, yet, due to several nuances, worth repeating as a whole).

$$x^v_{a,a'} \leq \sum_{t \in \mathcal{T}_a^{=v}} x^i_{a'} \qquad \forall a, a' \in \mathcal{A}, a \neq a' \, \forall v \in [u_{\max}], \tag{5.24}$$

$$\sum_{v=0}^{u_{\max}} x^v_{a,a'} = 1 \qquad \forall a, a' \in \mathcal{A}, a \neq a', \tag{5.25}$$

$$x^{-v}_{a,a'} \leq \sum_{t \in \mathcal{T}_a^{=-v}} x^i_{a'} \qquad \forall a \in \mathcal{A}, a \neq a' \, \forall v \in [u_{\max}], \tag{5.26}$$

$$x^0_{a,a} + \sum_{v=1}^{u_{\max}} x^{-v}_{a,a} = 1 \qquad \forall a \in \mathcal{A}, \tag{5.27}$$

$$y^v_{a,a'} \leq x^v_{a,a'} + x^{-v}_{a,a'} \qquad \forall a \in \mathcal{A}, a \neq a' \, \forall v \in [u_{\max}], \tag{5.28}$$

$$\sum_{v=1}^{u_{\max}} y^v_{a,a'} \leq 1 \qquad \forall a, a' \in \mathcal{A}, a \neq a'. \tag{5.29}$$

To explain the above ILP formulation, let us fix a pair $a, a' \in \mathcal{A}$ of distinct agents. Condition (5.24), for some utility value $v \in [u_{\max}]$ ensures that some resource of utility v for agent a can be a candidate for the best-case choice resource if it actually is in the bundle of a'. Condition (5.25) ensures that we are considering at most one such a candidate for the best-case choice resource and, in case such a candidate does not exist, Condition (5.25) ensures that we encode this situation properly by setting $x^0_{a,a'}$ to 1. For some agent $a \in \mathcal{A}$, Conditions (5.26) and (5.27) work analogously to Conditions (5.24) and (5.25) but for best-case choice resources among those that are assigned to a initially. Finally, Conditions (5.28) and (5.29) assume that for agents a and a' at most one "best-choice resource" to neglect is chosen.

To conclude the ILP model, we add the following constraint that guarantees that allocation π is EF1:

$$\sum_{t \in \mathcal{T}} x^i_a u_a(t) \geq \sum_{t \in \mathcal{T}} x^i_{a'} u_a(t) - \sum_{v=1}^{u_{\max}} v y^v_{a,a'} \qquad \forall a, a' \in \mathcal{A}, a \neq a'. \tag{5.30}$$

To show that Condition (5.30) is correct, we first assume that allocation π is EF1 and we fix (without loss of generality) a pair $a, a' \in \mathcal{A}$ of distinct agents and (again without loss of generality) consider the situation that a is comparing its bundle to the one of a'. If a is not envious, then we do not even need to

artificially neglect a resource. Thus, all conditions of the model are met with all variables set to 0; except for the special variables $x^0_{a,a}$ and $x^0_{a,a}$ that have to be set to 1. If a was initially envious, then let r be a resource of type $t \in \mathcal{T}$ with utility value v or $-v$ for a such that, after neglecting r, a is not envious. In such a case, we set $y^v_{a,a'}$ to 1 fulfilling Condition (5.30). Depending on whether $r \in \pi(a)$ we either set $x^{-v}_{a,a'}$ or $x^v_{a,a'}$ to 1—since r is either in the bundle of a or a' we definitely can do so without invalidating Conditions (5.24) and (5.26).

For the reverse direction, let us assume that the model is feasible with some solution S. Without loss of generality, we, again, fix a pair $a, a' \in \mathcal{A}$ of distinct agents and (also without loss of generality) consider the situation that a is comparing its bundle to the one of a'. We first focus on Condition (5.30). If, for all $v \in [u_{\max}]$, $y^v_{a,a'} = 0$, then agent a does not envy a'. Otherwise, there is (at least) one resource with some utility v or $-v$ that can be neglected, which will lead to making a non-envious towards a'. Observe that due to Condition (5.29) there is exactly one such value v. Moreover, Condition (5.28) guarantees that at least one resource with utility value v or $-v$ for a has been selected by the variables of form $y^v_{a,a'}$ and $y^{-v}_{a,a}$. Due to the semantics of these variables described earlier, it guarantees that indeed there is a resource $r \in \mathcal{R}$ (assuming allocation π) meeting the criterion of EF1 from Definition 5.13.

5.4.3 Graph Envy-Freeness

Another way of relaxing the concept of envy-freeness is to restrict the ability of agents to feel envy. One such relaxation is, studied in Chapter 4, graph envy-freeness. Herein, agents are embedded into a social network and they can be envious towards their neighbors in this social network. Below we recall Definition 4.8.

Definition 5.14. Let $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ be a directed graph, called an *attention graph*, representing a social network over the agents (i.e., the agents are the vertices of $\mathcal{G}$). Allocation π is *(weakly) graph-envy-free* if for each pair of (distinct) agents $a_1, a_2 \in \mathcal{A}$ such that $a_2 \in N^+(a_1)$ it holds that $u_1(\pi(a_1)) \geq u_1(\pi(a_2))$. By replacing the weak inequality in our criterion with a strict inequality, we obtain the definition of a *strongly graph-envy-free* allocation.

Observe that this rather general idea of narrowing down the possibility of envying according to some graph can also be easily applied to EF1, EFX, and similar concepts whose core depends on comparing two (or even more) agents. Furthermore, this kind of graph-based concepts are equivalent to their "bare" counterparts when the social network is a complete (bi-directional) graph.

Indeed, a similar observation applies to our ILP formulations. We do not give a specific ILP formulation for graph envy-freeness because, in fact, it is possible to combine all EF, EF1 and EFX with a graph and thus obtain the corresponding graph versions of these notions.

Observe that the graph does not *add* specific constraints. In fact, it always works as a "filter" for the constraints that were already presented in the devised ILP formulations; for example, if the graph consists of all possible arcs except for an arc from some agent a to agent a', then it suffices to remove all constraints comparing this particular pair of agents (note that in the presented models the order of agents in a pair matters). Thus, to construct ILP formulations for the respective graph-versions of various envy-freeness concepts, one has to apply the corresponding, already presented ILP formulation and delete some constraints depending on the attention graph. Naturally, deleting constraints cannot increase the number of variables of any ILP formulation, so the resulting ILP formulation would also meet the criteria of Theorem 5.6.

5.5 Experimental Evaluation

The theoretical upper bounds shown by Theorem 5.4 are purely of classification nature and cannot be used to derive any reasonable running time upper bounds in reality.

Let us, for example, analyze the theoretical worst-case running time of an algorithm straightforwardly based on the proof of Theorem 5.4. Recall that Proposition 2.1 says that one can solve an ILP instance $\mathcal{I}$ with t variables in $O\left(t^{2.5t+o(t)} \cdot \langle \mathcal{I} \rangle\right)$ time (where $\langle \mathcal{I} \rangle$ is the input size). Considering n agents, maximal utility $u_{\max}$, and h resource types, our final integer linear program (as depicted in the proof of Theorem 5.4) has $t \geq (nu_{\max})^{n^2 h}$ variables. Thus, if we have ten agents and maximum utility ten, then, substituting h by its upper bound $n^u_{\max}$, already $t \leq 100^{10^{12}}$ becomes enormous and the theoretical running time is far more than t^t; such an instance is clearly unsolvable in practice. Even if we assume that, as its likely in practice, there are much fewer resource types than $n^u_{\max} = 100$, say 10, then we arrive at $t \geq 100^{1000}$; which again yields an instance unsolvable in practice (recall, that the running time is at least t^t). So, clearly, a naïve algorithm directly applying Theorem 5.4 would not work in practice.

5.5.1 Towards Practical Implementation

The major reason why the naïve implementation of the proof of Theorem 5.4 would be practically computationally infeasible lies in the size of the set $\mathcal{D}$ of potential changes that can lead to a dominating allocation. Indeed, the integer linear program depicted in Section 5.3.3 contains a separate collection of constraints and variables for each element of $\mathcal{D}$. As a result, the number of variables of the integer liner program increases exponentially.

Intuitively, the ILP from Section 5.3.3 is a collection (conjunction) of multiple ILPs. One of them "finds" an envy-free allocation and all others ensure that there is no change that dominates this allocation. The problem of finding a Pareto-efficient envy-free allocation is Σ_2^{P}-hard [BL08]. Without going into formal details, it means that solving multiple ILPs in unavoidable to tackle the problem. However, instead of solving them at one shot—like the ILP from Section 5.3.3 does—we resort to the paradigm of separation subroutine [GLS81, Sch86]. Therein, one starts with solving a small ILP, and then one iteratively uses another ILP to improve the original solution. If the original solution can be improved, then the original ILP is augmented with new constraints preventing it from obtaining the original solution. Next, we demonstrate that this high-level description of the subroutine separation technique suits well for our use case.

We start with the following ILP, presented using the predicates introduced in Section 5.3,

$$\varphi \coloneqq (\exists \mathbf{x})\,(\mathrm{allocation}(\mathbf{x}) \wedge \mathrm{EF}(\mathbf{x}))\ .$$

Assuming that $\boldsymbol{\pi}$ is an allocation found by solving φ, we then solve the Allocation Dominance problem for allocation $\boldsymbol{\pi}$. If the answer is negative, then $\boldsymbol{\pi}$ is envy-free and Pareto-efficient. Otherwise, when $\mathbf{y}$ dominates $\boldsymbol{\pi}$, let $\boldsymbol{\Delta} \coloneqq \mathbf{y} - \boldsymbol{\pi}$ be the *improving change* computed by the Allocation Dominance ILP. We employ the same technique as the one presented in Section 5.3.3 and, defining $\#_{\max} \coloneqq \max_{t \in \mathcal{T}} \#(i)$, we modify φ adding the following constraints:

$$x_a^t + \Delta_a^t \leq -1 + (1 - z_a^t)(\#_{\max}) \qquad \forall a \in \mathcal{A}\, \forall t \in \mathcal{T}, \tag{5.31}$$

$$x_a^t + \Delta_a^t \geq -(\#_{\max}) + \hat{z}_a^t(2\#_{\max} + 1) \qquad \forall a \in \mathcal{A}\, \forall t \in \mathcal{T}, \tag{5.32}$$

$$\sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} \left(z_a^t + \hat{z}_a^t\right) \geq 1\,. \tag{5.33}$$

Observe that the improving change is a constant in the updated φ. We repeat the above procedure, this time starting with the updated φ until either a subsequent φ yields an allocation $\mathbf{x}$ that is not Pareto-dominated or φ becomes

infeasible. In the former case, allocation $\mathbf{x}$ is envy-free and Pareto-efficient; in the latter case, an envy-free and Pareto-efficient allocation does not exist.

The main idea behind the above approach is basically the same as that of Section 5.3.3; the only difference is that here we compute elements of $\mathcal{D}$ dynamically and we add them "on request." In the worst-case, it still can happen that we have to compute all elements of $\mathcal{D}$. Hence, our technique might be successful only if, on average, there are relatively few changes in $\mathcal{D}$ that actually lead to Pareto-dominating allocations (or that certify that all envy-free allocations are Pareto-dominated). Indeed, in the remainder of this section, we show that for real-world data it actually frequently happens that despite enormous theoretical running time, our dynamic approach allows for finding envy-free and Pareto-efficient allocations quickly.

5.5.2 Experimental Setup

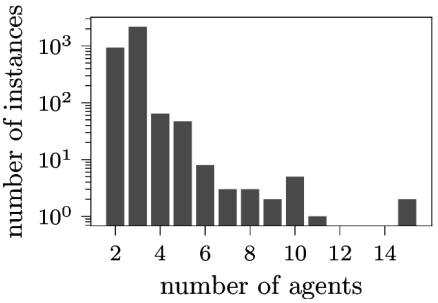
We implemented the aforementioned procedure and then tested it on real-world data gathered by the free, publicly-accessible, online service spliddit.org [Pro+20]. The service, after being fed with a list of indivisible resources, agents and utilities that the agents give to the resources, computes an allocation that is guaranteed to be envy-free up to one good [GP15]. The data set, which we subsequently refer to as the *spliddit data set*, was shared with us by the authors of the service: Ariel Procaccia, Jonathan Goldman, Nisarg Shah, and David Kurokawa.

In our implementations we used C++ and the IBM ILOG CPLEX Interactive Optimizer (version 12.9.0.0) for solving the integer linear programs. The specification of the machines used for our experiments is provided in Section 2.7.

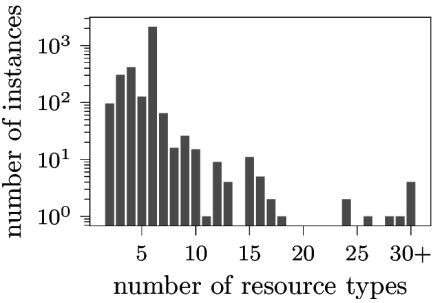
Data

The spliddit data set consists of 3244 instances with up to 15 agents and up to 85 resource types. Before we briefly describe the data set, we refer to the more detailed statistics depicted in a series of charts in Figure 5.3.

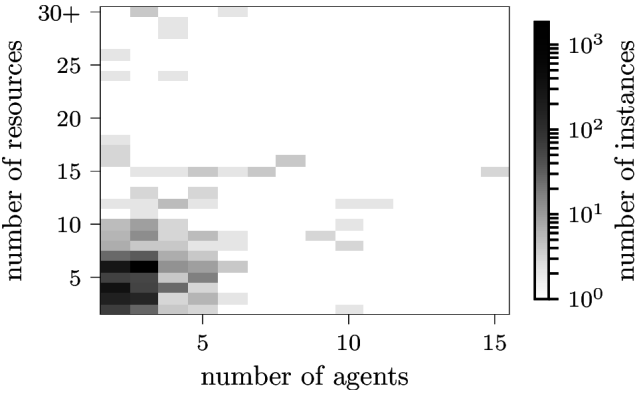
By design, spliddit.org requires all necessary input data to be manually provided via the web graphic interface—it is not possible to feed the service with data in bulk in any form. This observation justifies the following characteristic feature of the spliddit data set. Overall, as it can be noticed in Figure 5.3, there is a large majority of small instances that contain few agents and few resource types. However, as already mentioned at the beginning of this section, even for such small numbers our algorithm, if implemented naïvely, would run extremely long. Importantly, the spliddit data set does not contain any resources that



(a) Counts of instances with different numbers of agents.



(b) Counts of instances with different numbers of resource types.



(c) Counts of instances with particular numbers of resource types and agents.

Figure 5.3.: Plots presenting the statistics of the spliddit data set.

fairness criterion	min [s]	avg [s]	max [s]
EF	0.05	0.01	2.02
EFX	0.66	0.03	73.47
EF1	0.67	0.03	22.67

Figure 5.4.: The minimum, average, and maximum running time required to find a Pareto-efficient allocation meeting a given fairness criteria or decide that such an allocation does not exist.

have negative utility values. This is a consequence of the construction of the online interface that does not allow to choose negative utilities.

We also mention that in the instances of the spliddit data set, each resource comes in one copy. This is expected, since the service allows for submitting evaluations privately by each agent (participant). Thus, it is highly unexpected that all participants provide exactly the same evaluation of all resources.

5.5.3 Experimental Results

Our goal was to assess whether our implementation is fast enough to be used in practice. The running times we obtained while experimenting with the spliddit data set suggest that our approach is indeed appropriate for real-world applications that involve human agents.

As depicted in Figure 5.4, we were always able to find an envy-free and Pareto-efficient allocation or decide that such an allocation does not exist within two seconds. In at most 75 seconds, we were able to compute allocations that are Pareto-efficient and envy-free up to any resource (EFX) using the ILP model devised in Section 5.4.1.

In the discussed experiments, we took an advantage of setting an optimization goal for an ILP. We set the goal so that we sought allocations that are Pareto-efficient, fair (with respect to the three fairness concepts we experimented with), and additionally maximize the total sum of utilities that the agents get from their own bundles—a measure frequently called *social welfare*. We observed that adding this optimization goal allowed to significantly (for some instances even more than ten times) speed up the algorithm. Intuitively, the speed-up comes from the observation that if no resource has negative utility (for any agent), then each allocation maximizing social-welfare is also Pareto-efficient (the opposite is not true). For clarity, in Figure 5.4, we decided not to present the running times for the case for which we did not use the optimization goal.

It turned out that even though EFX is a relaxation of envy-freeness, the computation took considerably more time. We conjecture that this is due to the fact that the classical envy-freeness together with Pareto-efficiency form a very discriminative collection of properties. So, on the one hand, if allocations meeting these properties exist, then one needs to look for such allocations in a relatively small search space, which can be done efficiently. On the other hand, if such allocations do not exist, then finding this out is not computationally demanding either. For comparison, we also ran our benchmark for seeking allocations that are Pareto-efficient and envy-free up to one resource. On average, the observed running times, depicted in more details in Figure 5.5, were very similar to those of seeking EFX and Pareto-efficient allocations. However, the maximum running time, compared to that of finding a Pareto-efficient and EFX allocation, dropped roughly three times. This phenomenon most likely comes from the fact that an allocation that EFX is also, by definition, EF1 but not necessarily the other way round.

5.6 Conclusions

We have presented the fairly general Theorem 5.6 for scenarios of high-multiplicity fair resource allocation of indivisible resources. The theorem exploits structural properties of ILP formulations of the studied notions of both allocation fairness and Pareto-efficiency. Remarkably, an important feature of Theorem 5.6 is that it covers fair allocation scenarios with indivisible resources having only negative utilities (so-called indivisible chores allocation [Azi+19]) and scenarios where an indivisible resource could be assigned a negative utility for one agent and a positive utility for another agent (so-called allocation of mixed manna [AW19]).

We believe that Theorem 5.6 substantially contributes to our understanding of the differences (and similarities) between the studied fairness concepts. All studied fairness concepts admit ILP formulations with a number of variables upper-bounded by the number of agents plus the maximum absolute value of utility to apply Theorem 5.6. However, some of them are significantly more complicated than the others, indicating that it might be demanding to write an ILP formulation that meets the criteria of Theorem 5.6.

In our experiments, we confirmed that our approach is applicable for real-world problems by obtaining practically feasible running times on the data from spliddit.org. Thus, we believe we have brought real added value not only from the theoretical perspective but also from the practical one.

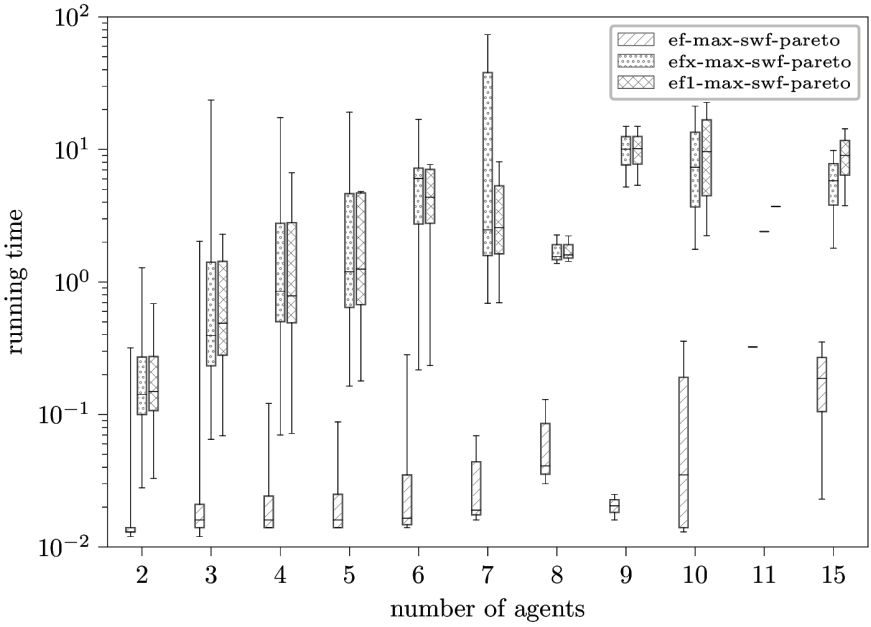


Figure 5.5.: Comparison of running times (in seconds) of finding Pareto-efficient allocations meeting different notions of fairness for various numbers of agents. Each box shows quartiles of the collected data with the median depicted by the middle bar. The so-called whiskers around the boxes represent the maximum and minimum.

It is worth pointing out that the allocation fairness concepts discussed in Section 5.4 do not form an exhaustive list of what can be found in the literature. The intended meaning is to provide some examples of what is possible. We believe that a significant number of already studied notions of allocation fairness admit ILP formulations compliant with Theorem 5.6 (note that Bredereck et al. [Bre+19b] provide ILP formulations for a few more fairness notions).

As for future work with respect to theoretical studies, it is widely believed that general statements like in Theorem 5.6 are highly beneficial classification results; however, algorithms designed for specific problems usually outperform this approach both in theory and practice. This leads us to the open questions of providing an algorithm for EEF–Allocation with better running time or running time lower bounds.

The practical part of our work brings up at least two challenges. First, we have conducted our test on data that, in fact, did not have resources coming in high-multiplicity. We are not aware of any real-world data that would feature the high-multiplicity regime. Thus, to draw valuable insights from testing the algorithm in the high-multiplicity scenario, it is desirable to first gather data that would inherently have resources coming in many copies. Apart from a natural way which is to collect real-world data, this could also be achieved by coming up with appropriate data generation models. Second, showing that we are able to solve fair allocation problems relatively effectively calls for more insightful experiments leading to, for example, checking robustness of solutions, finding out qualitative differences between allocations, or learning important features that could give hints for designing faster, specialized algorithms.

Part II

Multiwinner Voting. Dealing with Collective Sets of Resources

In Part II we shift our focus from the traditional resource allocation scenario considered in Part I—where agents own resources privately—to a variant where the goal is to find a best collection of resources that are shared by all agents and thus, in some sense, satisfy them.

As a natural consequence, since here we have no individual bundles but rather a single, collective set of resources, the nature of addressed questions changes significantly. Observe that we are no longer concerned with envy between agents because all agents share the same resources, implying that there are no bundles. Thus, formally, we have an ideal situation where all envy (according to the definitions we used in Part I) disappears. This perspective immediately raises a question about a middle ground between the two extremes of private bundles of resources and a collective set of resources. Indeed, such scenarios, implemented by allowing sharing resources between agents, recently have also been studied from the computational perspective [SS19]. In Part II, however, we are only interested in the well-established (as we will explain in the next paragraph) extreme case of a *single* collective set of resources.

As already mentioned, studying the case of a collective set of resources leads to asking fairly different questions than those in the classical fair allocation scenarios. Indeed, for a single set of commonly shared resources, instead of discussing envy, we are rather concerned with how "satisfying" a particular collection of resources is for each agent, and how to choose such a set of resources satisfying all agents most. Ideally, we would like to achieve a certain

level of satisfaction using as few resources as possible or a fixed number of resources. This goal occurs naturally whenever we face a situation that we have limited space for resources to keep or whenever each resource comes at some (not necessarily tangible) cost. The second described variant—selecting a fixed number of collective sets of resources—is actually equivalent to multiwinner voting intensively studied in social choice during the last years (we refer to the book chapters [Fal+17, FR15, LX15] for a comprehensive list of multiwinner voting literature). Indeed, by representing voters by agents and resources by candidates, we exactly arrive at multiwinner voting, where voters specify preferences over (possibly sets of) candidates and the task is to select a "best" set of candidates—a committee. Thus, from now on, we turn to the domain of multiwinner elections and use the terms "voters" and "candidates" instead of, respectively, "agents" and "resources."

We devote Part II to studying multiwinner elections. However, this time instead of focusing on how to achieve a "best" outcome (which by now is a relatively well-studied problem [Elk+17, Fal+19]), we rather investigate how hard it is to change some, already existing, outcome by changing the preferences of the voters. This question, formally relating to the robustness (or stability) of an outcome, is especially relevant since multiwinner voting is extensively used in the real world (from political elections through selecting nominees for prizes such as The Oscars [Aca19] to hiring employees by HR departments [Ide20]). However, before we start studying the computational complexity of changing a multiwinner election outcome (Chapter 7) and of strategically influencing a multiwinner election by a group of voters (Chapter 8), in the following Chapter 6 we provide a short primer on multiwinner voting.

CHAPTER 6

Elections, Multiwinner Voting Rules, and Election Generation

In this chapter, we lay the formal foundations for investigating multiwinner elections from the practical perspective as well as for studying them through the theoretical lens.

6.1 Elections and Multiwinner Voting

The intuitive goal of these elections is to select a most suitable (with respect to a given goal) set of candidates of a given size based on the voter preferences. Naturally, voter preferences can be expressed in various ways. In contrast to indivisible resource allocations (which we considered in Part I of the thesis) where preferences are usually expressed cardinally, in the area of voting preferences are frequently expressed as rankings from the most to the least desirable candidate. This intuitive description can leads to the following Definition 6.1.

Definition 6.1. An *election* $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ consists of a set $\mathcal{C} = \{c_1, c_2, \dots, c_m\}$ of *candidates* and of a collection $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ of *voters*, where each voter v_i is associated with a total order $\succ_i$ over the candidates in $\mathcal{C}$ sometimes called a *vote* or a *ranking*. For a vote $v_i \in \mathcal{V}$, the *position* $\mathrm{pos}_{v_i}(c) \in [m]$ of a candidate $c \in \mathcal{C}$ is 1 if it is the most preferred according to $\succ_i$, it is 2 if it is the second most preferred according to $\succ_i$, and so on.

Instead of $\succ_i$, we write $\succ$ whenever the voter v_i under consideration is clear from the context. Throughout the thesis, for two disjoint sets A and B of candidates, sometimes, when specifying a preference order, we write $A \succ B$ to denote the fact that each candidate in set A is preferred to each candidate in set B, but the particular order of the candidates within these sets is irrelevant for the discussion. See Example 6.1 for an illustration of Definition 6.1 and the conventions introduced above.

> **Example 6.1.** Let $\mathcal{C} = \{a, b, c, d\}$ be a set of four candidates and let $D = \{b, d\} \subseteq \mathcal{C}$. Consider an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$, where $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ is a set

> of four voters with the following total orders (rankings) over the candidates in $\mathcal{C}$:
>
> $$v_1\colon a \succ c \succ b \succ d,$$
> $$v_2\colon a \succ c \succ d \succ b,$$
> $$v_3\colon d \succ a \succ c \succ d, \text{ and}$$
> $$v_4\colon c \succ D \succ a.$$
>
> Now, the position of candidate a in vote v_3 is 3, whereas candidate d is ranked on top by voter v_3. Since vote v_4 is either $c \succ b \succ d \succ a$ or $c \succ d \succ b \succ a$, we can only say that $\mathcal{V}_4$ ranks c first and a last.

Definition 6.1 of elections does not contain any process that might be used to settle the outcome of an election. Thus, to determine the winner(s) of an election a voting rule has to be applied.

Definition 6.2. A *multiwinner voting rule* $\mathcal{M}$ is a function that, given an election $(\mathcal{C}, \mathcal{V})$ and a positive integral *committee size* $k \leq |\mathcal{C}|$, outputs a set $\mathcal{M}(\mathcal{E}, k)$ of size-k subsets of $\mathcal{C}$ referred to as the *winning committees*. If there are multiple winning committees, they are considered to be *tied* for victory.

One may argue that the above definition should rather use the name multiwinner voting correspondence for $\mathcal{M}$ (instead of multiwinner voting rule) because it returns a set of tied committees instead of a single committee (see, for example, the discussion on ties by Obraztsova, Zick, and Elkind [OZE13]; a textbook chapter on elections [BR15]; or a work by Barberà, Sonnenschein, and Zhou [BSZ91] where a voter instead of ordering the candidates was assumed to order all subsets of candidates of a given size). However, the phrasing of Definition 6.2 is now well-established [Bra+16, End17] and without doubts most frequently used in the literature.

6.1.1 Committee Scoring Rules

To give some intuition on multiwinner voting rules, we devote this section to introduce those of them, that are used in both Chapters 7 and 8. Intuitively, for some election and a requested committee size k, the rules in the family of committee scoring rules assign a score to each committee consisting of k candidates and they select those committees that obtain the highest score. We formally describe the above procedure and then provide a toy example as an illustration.

A fundamental part of committee scoring rules is a scoring function associating scores with positions in votes.

Definition 6.3. A *scoring function* for m candidates is a function $\gamma_m \colon [m] \to \mathbb{Q}$ that associates each candidate-position within a given vote a score.

Below, we define some scoring functions frequently studied in the literature as well as used in practice and afterwards we give an example to demonstrate their application.

Definition 6.4. Let m be the number of candidates. The *Borda scoring function* function $\beta_m(i) \coloneqq m - i$ associates to a candidate at some position $i \in [m]$ in a vote a score equal to the number of candidates ranked worse. For some $t \in [m]$, the *t-Approval* scoring function $\alpha_t(i)$ is 1 if $i \leq t$ and 0 otherwise. We refer to the special case α_1 as the *Plurality* scoring function.

For reasons of convenience, we frequently use the term "Approval score" instead of α_t-score and the term "Borda score" instead of β_m-score if it does not lead to ambiguity. Having defined scoring functions, we proceed with defining the score of a candidate in an election.

Definition 6.5. For a scoring function γ_m, the *γ_m-score* of a candidate $c \in \mathcal{C}$ in an m-candidate election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ is defined as $\gamma_m\text{-score}_{\mathcal{E}}(c) \coloneqq \sum_{v \in \mathcal{V}} \gamma_m(\text{pos}_v(c))$.

Depending on which scoring functions are used and how exactly scores of committees are computed, we obtain different multiwinner rules.

The Single Non-Transferable Vote voting rule (SNTV) outputs the committees whose members have the highest sum of Plurality scores. To select a committee of size k, one can also use the k-Borda voting rule. This rule works analogously to SNTV, however, instead of using the Plurality scoring function, it applies the respective Borda scoring function. Yet another analogous committee scoring rule is t-Approval using the t-Approval scoring function. Note that t-Approval does not necessary use a value of t that is equal to k. Indeed, whereas in Chapter 7 we focus entirely on the case of $t = k$, in Chapter 8 we also consider scenarios where it is not the case. Due to the similarities of above-mentioned rules, we deal with them collectively in Definition 6.6.

Definition 6.6. Consider some election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ with m candidates, some committee size k, and a committee $S = \{c_1, c_2, \ldots, c_k\} \in \mathcal{C}$. The *score* of committee S is:

1. α_1-score$(S) \coloneqq \sum_{c \in S} \alpha_1$-score$(c)$ for the *Single Non-Transferable Vote* voting rule;
2. α_t-score$(S) \coloneqq \sum_{c \in S} \alpha_k$-score$(c)$ for the t-*Approval* voting rule with a special case of $t = k$ referred to as k-*Approval*; and
3. β_k-score$(S) \coloneqq \sum_{c \in S} \beta_k$-score$(c)$ for the k-*Borda* voting rule.

The SNTV, k-Borda, and t-Approval rules select those committees that maximize the respective committee scores.

In passing, we note that t-Approval related rules come in the literature under various names such as *Limited Voting* [KM15], *Constant Scoring Rules* [GF81], t-*Bloc* [Mei+08], and, for the case of $t = k$, *Bloc* [DD16]. The following example showcases the rules from Definition 6.6 and displays their differences.

Example 6.2. Consider the following election with six candidates a to f:

$$\begin{aligned} v_1 &\colon a \succ d \succ e \succ b \succ c \succ f, \\ v_2 &\colon a \succ d \succ e \succ b \succ f \succ c, \\ v_3 &\colon b \succ c \succ e \succ d \succ f \succ a, \\ v_4 &\colon b \succ d \succ f \succ d \succ e \succ a, \text{ and} \\ v_5 &\colon e \succ a \succ c \succ b \succ f \succ d. \end{aligned}$$

We show which committees of size $k = 2$ are selected by SNTV, k-Approval, 3-Approval, and k-Borda,. Since SNTV is exactly the same as 1-Approval and k-Approval in our case is 2-Approval we actually show outcomes of under all of 1-Approval, 2-Approval, and 3-Approval.

The committee $\{a, b\}$ is selected as a unique winning committee under 1-Approval since both a and b are ranked on the first position by exactly two voters, and there is no other candidate with α_1-score at least two. Moving to k-Approval, it turns out that the unique winning committee is $\{a, d\}$ because candidate b whose α_2-score is two is outperformed by candidate d who appears three times at the first two positions of the voter rankings. Analogously, under 3-Approval, we again obtain a different outcome consisting of two tied committees $\{e, a\}$ and $\{e, d\}$.

To compute the outcome of k-Borda, we first have to compute Borda scores of the candidates. To demonstrate how to compute it, let us consider candidate a. Since a is two times at the first position and on the last position

> and once on the second position, the Borda score of a is $2 \cdot 5 + 4 + 2 \cdot 0 = 14$. One can easily verify that the Borda score of b, being 16, is the highest, that the Borda score of e, being 15, is second highest, and that no other candidate exceed the score of 14. Thus, the winning committee under k-Borda is $\{b, e\}$ and is distinct from the outcomes of all other demonstrated rules.

All rules defined above—SNTV, t-Approval, and k-Borda—are examples of committee scoring rules for which finding the winners can be done in polynomial time assuming the number of winning committees is not exponential with respect to the input size [Elk+17, Fal+19, SFS19].

6.2 Generating Synthetic Election Data

The Impartial Culture and Mallows distributions lie at the foundations of probably most common election generation models in the literature on experimental election study [BR16]. Here we provide their definitions and brief intuitive explanations as we also adapt them for our experiments; for a detailed discussion and literature overview we refer to a book chapter by Boutilier and Rosenschein [BR16]. For brevity, we interchangeably use words "distribution" and "model"; for example, we refer to a model that generates elections according to the Impartial Culture distribution as the Impartial Culture model.

We start with the simpler (and also less expressive) of the two models we use—the Impartial Culture distribution.

Definition 6.7. In the *Impartial Culture distribution*, for a set $\mathcal{C}$ of candidates, each ranking over the candidates in $\mathcal{C}$ is equally likely.

Despite of being appealing because of its simplicity and intuitiveness, Impartial Culture is generally believed not to reflect real-life elections well; in particular, it cannot model voters that share a very similar opinions on the candidates. Thus, to better explore the space of possible elections, there is a need of a more expressive model—one of them being the Mallows model. A rough intuition behind the Mallows model is that there is a given central ranking and the more swaps are necessary to modify some preference order v to become this central one, the less probable it is to draw v; in particular, the central order is the most probable one to be generated.

Definition 6.8. Let v_0 be the *central ranking* over a fixed set of candidates, ϕ be a *dispersion parameter*, d be a distance measure between two rankings.

Then, in the *Mallows distribution*, the probability of a given ranking v is

$$\frac{\phi^{d(v_0,v)}}{Z} \quad \text{where} \quad Z = 1 \cdot (1+\phi) \cdot (1+\phi+\phi^2) \cdots (1+\cdots+\phi^{m-1}).$$

Note that the normalization constant Z in Definition 6.8 is independent of $\mathcal{V}_0$. Observe that for $\phi = 1$, the Mallows model becomes equivalent to the Impartial Culture model and for $\phi = 0$ it draws the central ranking $\mathcal{V}_0$. For generating elections in our work, as the distance measure from Definition 6.8, we used the Kendall tau distance that measures the number of swaps of adjacent candidates needed to make two measured rankings identical. For reasons of simplicity, in the definition below we abstain from providing a very formal definition as in our experiments, to generate elections, we used external generators provided by the PrefLib library [MW13].

Definition 6.9 ([Ken38])**.** The *Kendall tau distance* between two rankings v and v' is equal to the number of swaps of adjacent candidates that are necessary to transform v into v'.

In our experiments, we also use a so-called Mixed Mallows model, which is a combination of two Mallows models (with independent central orders and values of parameter ϕ) *from which* one samples rankings with respect to some given distribution.

Definition 6.10. Let M_1 and M_2 be two Mallows distributions with probabilities P_v^1 and P_v^2 of a ranking v; and let p the probability of drawing an order from model M_1. The *Mixed Mallows distribution* (over distributions M_1 and M_2) provides some ranking v with probability $p \cdot P_v^1 + (1-p)P_v^2$.

Intuitively, Mixed Mallows allow for exploring the space of possible elections by providing bimodal distributions of votes.

CHAPTER 7

Robustness of Multiwinner Voting Rules

We investigate how robust the results of committee elections are to small changes in the input preference orders, depending on the voting rules used. We focus on the problem of computing the smallest number of swaps that lead to changing the election outcome. We show that this problem is typically NP-hard, but we also spot fixed-parameter tractability for natural parameters such as the number of voters or the number of candidates. Finally, for a number of rules, we assess experimentally the average number of random swaps necessary to change the election result.

7.1 Introduction

We study how multiwinner voting rules—that is, procedures used to select fixed-size committees of candidates—react to (small) changes in the input votes. We are mainly interested in the complexity of computing the smallest modification of the votes that affects the election outcome. Before we present our ideas formally, we discuss them on the intuitive level in the following example.

Consider a research-funding agency that needs to choose which of the submitted project proposals to support. The agency asks a group of experts to evaluate the proposals and to rank them from the best to the worst one. Then, the agency uses some formal process—here modeled as a multiwinner voting rule—to aggregate these rankings and to select k projects to be funded. (In practice, the agency would, likely, use a two-stage process and the selected projects would be sent for further, more detailed, evaluation; very roughly put, this is how the National Science Centre (NCN) in Poland operates.) Imagine that one of the experts realized that, instead of ranking a proposal A better than a proposal B, he or she should have given the opposite opinion. Would such a "mistake" influence the result of the process? Depending on the multiwinner voting rules used, it may not affect the results, it may cause a minor change or a great change in the result. Thus, the agency prefers to avoid such situations by making the process more *robust* against such unintentional changes, as a result increasing confidence about the final selection. So, the agency would want to be able to compute the smallest number of swaps that change the result. In

cases where this number is too small, the agency could possibly fight this issue by inviting more experts to gain more stable results.

Below we provide a slightly more formal introduction. A multiwinner voting rule (recall Chapter 6) is a function that, given a set of rankings of the candidates and an integer k, outputs a family of size-k subsets of the candidates—the winning committees. We consider the following two issues (for simplicity, below we ignore ties and assume to always have a unique winning committee):

1. We say that the *robustness radius* of an election $\mathcal{E}$ (for committee size k) under a multiwinner rule $\mathcal{M}$ is the smallest number of swaps of adjacent candidates which are necessary to change the election outcome. We ask for the complexity of computing the robustness radius (referred to as the Robustness Radius problem) under a number of multiwinner rules. This problem is strongly related to the Margin of Victory [BST19, Car11, Mag+11, Xia12] and Destructive Swap Bribery problems [EFS09, SYE13]. Furthermore, our work follows up on the study of Shiryaev, Yu, and Elkind [SYE13], who considered the robustness of single-winner rules.

2. We ask (for a given voting rule) how many random swaps of adjacent candidates are necessary, on average, to move from a randomly generated election to one with a different outcome. We assess this kind of robustness of our rules experimentally for elections generated according to the Impartial Culture, Mallows, and a mixture of two Mallows models (see Section 6.2 for definitions), and for real-world data available in PrefLib [MW13].

There is quite a high number of multiwinner rules. We consider only some of them, selected to represent a varied set of ideas from the literature, ranging from variants of scoring rules, through rules inspired by the Condorcet criterion, to the elimination-based STV rule.

We show that the Robustness Radius problem tends to be NP-hard (sometimes even for a single swap). We seek fixed-parameter tractability results with respect to natural parameters to circumvent this hardness. For example, we find several algorithms that yield fixed-parameter tractability with respect to the number of voters (these algorithms are useful, for instance, for scenarios with few experts, such as in our introductory example). See Table 7.1 for an overview of our theoretical results. We mention that Misra and Sonar [MS19] followed up on our results and, in particular, have considered several variants of the Chamberlin–Courant rule and certain nearly-structured preference domains.

Recently, Gawron and Faliszewski [GF19] applied our notions of robustness to the case of approval elections.

We furthermore perform an experimental evaluation of the robustness of our rules with respect to random swaps. We conclude that, on average, to change the outcome of an election, one needs to make the most swaps under the k-Borda rule, whereas STV and SNTV (Single Non-Transferable Vote) require fewest swaps to achieve this result.

This chapter is organized as follows. In Section 7.2 we provide the necessary background, including the definitions of the rules that we focus on. Then, in Sections 7.3 and 7.4, we introduce the Robustness Radius problem and study its computational complexity; in the former section we mostly focus on the classic complexity, whereas in the latter we provide several FPT algorithms. In Section 7.5 we describe our experiments. We conclude our findings in Section 7.6.

7.2 Preliminaries

We first describe the voting rules on which we focus and then we explain our notion of robustness radius and the corresponding computational problem Robustness Radius.

7.2.1 The Chamberlin–Courant Rule

The Chamberlin–Courant rule [CC83] (as well as SNTV, k-Approval, and k-Borda; see Section 6.1.1) is an example of a committee scoring rule [Elk+17, Fal+19, SFS19], so it outputs the committees with the highest score computed according to some scoring function. The Chamberlin–Courant rule, however, computes the committee scores in a significantly different way than the rules from Definition 6.6, under which each candidate contributes a fixed score to all possible committee containing this candidate. In the Chamberlin–Courant rule, the score of a given committee depends on how much every voter is satisfied with the voter's highest-ranked candidate present in the committee under consideration; thus, the contribution of a candidate to the score of a committee depends also on the different candidates the committee contains. The satisfaction of a voter is measured using the Borda scoring function, β_m, which is the reason why we refer to the Chamberlin–Courant rule as β-CC.

[1] For STV there is a polynomial-time algorithm for computing a single winning committee, but deciding whether a given committee wins is NP-hard.

Voting Rule	Complexity of Robustness Radius	
SNTV, k-Approval, k-Borda (P)	P	(Th. 7.1)
k-Copeland$^\alpha$ (P)	NP-hard	(Cor. 7.3)
	FPT(m)	(Pr. 7.6)
	W[1]-hard(n)	(Cor. 7.10)
NED (NP-hard [Azi+17])	NP-hard	(Th. 7.2)
	FPT(m)	(Pr. 7.6)
	W[1]-hard(n)	(Cor. 7.10)
β-CC (NP-hard [BSU13, LB11, PRZ08])	p-NP-hard(B)	(Th. 7.4)
	FPT(m)	(Pr. 7.6)
	FPT(n)	(Th. 7.9)
STV (NP-hard[1] [CRX09])	p-NP-hard(B)	(Th. 7.5)
	FPT(m)	(Pr. 7.6)
	FPT(n)	(Th. 7.7)

Table 7.1.: Summary of our results. For each rule, we provide the complexity of its winner determination. The parameters m, n, and B mean, respectively, the number of candidates, the number of voters, and the robustness radius; p-NP-hard(B) means NP-hard even for constant B. We leave open the parameterized complexity of Robustness Radius for k-Copeland$^\alpha$ and NED with respect to B.

Definition 7.1. Consider an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ with m candidates, a committee size k, and a committee $S = \{c_1, c_2, \ldots, c_k\} \subseteq \mathcal{C}$. The *representative* $c^* \in S$ of a voter $v \in \mathcal{V}$ in S is the highest-ranked candidate in S according to v's ranking; that is, $c^* \coloneqq \arg\min_{c \in S} \mathrm{pos}_v(c)$. The *satisfaction* of voter v from S is the Borda-score (in v's ranking) of the voter's representative . The *dissatisfaction* of voter v from S is equal to $(m-1)$ minus the voter's satisfaction.

In plain words, the outcome of the β-CC rule consists of those committees that satisfy the voters the most or, equivalently, dissatisfy them the least. In Definition 7.2, we formally present the former variant.

Definition 7.2. Consider an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ with m candidates, a committee size k, and a committee $S = \{c_1, c_2, \ldots, c_k\} \subseteq \mathcal{C}$. Under the *Chamberlin–Courant* (β-CC) rule, the *score* of committee S is defined as β-CC-score$(S) \coloneqq$

$v_1\colon a \succ b \succ c \succ d,$
$v_2\colon c \succ b \succ a \succ d,$
$v_3\colon a \succ c \succ b \succ d,$
$v_4\colon b \succ a \succ c \succ d.$

	score of S	sat. (dissat.) v_1	v_2	v_3	v_4
$S = \{a, c\}$	11	3 (0)	3 (0)	3 (0)	2 (1)
$S = \{a, b\}$	11	3 (0)	2 (1)	3 (0)	3 (0)
$S = \{b, d\}$	8	2 (1)	2 (1)	1 (2)	3 (0)

Figure 7.1.: The election considered in Example 7.1 to demonstrate the β-CC rule.

$\sum_{v \in \mathcal{V}} \beta(\arg\min_{c \in S} \mathrm{pos}_v(c))$. The Chamberlin–Courant rule selects those committees that maximize the respective committee scores.

In the following Example 7.1, we present Definition 7.2 in action.

Example 7.1. Consider an election consisting of four candidates a, b, c, and d for which we apply β-CC to find a winning committee of size two. In Figure 7.1, we present four example voters, together with a table with the scores of several committees of size two and with the satisfaction and dissatisfaction values of the voters. Let us briefly analyze committee $S = \{a, c\}$ to demonstrate how to compute the scores presented in Figure 7.1. Candidate a is a representative of voters v_1, v_2, and v_4 because a is preferred to c by all of them. Both voters v_1 and v_3 rank candidate a on top; thus their satisfaction is 3, which in turn means that their dissatisfaction is 0. Voter v_4 ranks a at the second position which gives satisfaction 2 and dissatisfaction 1. Regarding voter v_2, its representative is candidate c that gives satisfaction 3 and dissatisfaction 0.

Eventually, the winners under β-CC are $\{a, c\}$ and $\{a, b\}$. We do not consider committees consisting of d because d cannot be a representative of any voter (d is always at the worst position). It can also be easily verified that committee $\{b, c\}$ obtains a worse score than that of the claimed winning committees.

Determining the winner under β-CC is well-known to be NP-hard [LB11, PRZ08] and W[2]-hard when parameterized by the committee size [BSU13]. Yet, there are many ways of dealing with this negative result, including FPT-algorithms for other parameters [BSU13], approximation algorithms [LB11, SFS15], algorithms for restricted domains [BSU13, Pet18, Sko+15], and heuristics [Fal+18b].

7.2.2 Condorcet-Inspired Rules

Other multiwinner rules include those based on the idea of a Condorcet winner. A candidate c is a Condorcet winner if for each other candidate d, *more than* a half of the voters prefer c to d. A candidate is a weak Condorcet winner if it is preferred to every other candidate by *at least* a half of the voters. Gehrlein [Geh85] generalized this notion to the multiwinner case as follows.

Definition 7.3. For an election, a committee is *Gehrlein strongly-stable* if every committee member is preferred to every nonmember by *more than* a half of the voters in the election. A committee is *weakly-stable* if every member is preferred to every nonmember by *at least* a half of the voters in the election.

We can naturally categorize multiwinner rules basing on whether they always output committees that coincide with all Gehrlein weakly- or strongly-stable committees of elections.

Definition 7.4. A multiwinner rule is *Gehrlein strongly-stable* if, for each possible election, it outputs exactly the Gehrlein strongly-stable committees whenever they exist. A multiwinner rule is *Gehrlein weakly-stable* if, for each possible election, it outputs exactly the Gehrlein weakly-stable committees whenever they exist.

We provide two exemplary representatives of Gehrlein strongly-stable and Gehrlein weakly-stable rules.

Definition 7.5 ([Coe04])**.** Consider an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ with m candidates, a committee size k, and a committee $S = \{c_1, c_2, \ldots, c_k\} \subseteq \mathcal{C}$. The *NED score* (Number of External Defeats) of committee S is the number of pairs (c, e) such that $c \in S$, $e \notin S$, and c is preferred to e by at least a half of the voters. The *NED rule* outputs the committees with the highest NED score.

In passing, we note that originally the definition of the NED rule [Coe04] used a "dual" definition of the NED score, and thus it was choosing committees whose NED score was the smallest. The NED rule is Gehrlein weakly-stable but not Gehrlein strongly-stable. This is intuitively expected, since NED grants points to committees for an external defeat "already" if a member of a committee is preferred by a half of the voters to some nonmember. This stands in contrast to Gehrlein strongly-stable committees, in which each member is preferred to every nonmember by more than a half of the voters. The next rule we define, k-Copeland$^\alpha$, is Gehrlein strongly-stable but not Gehrlein weakly-stable.

v_1: $a \succ b \succ d \succ c$,
v_2: $c \succ b \succ a \succ d$,
v_3: $a \succ c \succ b \succ d$,
v_4: $b \succ a \succ d \succ c$.

	NED score
$\{a, b\}$	4
$\{a, c\}$	4
$\{b, c\}$	3
$\{d, \bullet\}$	≤ 3

	2-Copeland$^{0.5}$ score
a	2.5
b	2
c	1
d	0.5

Figure 7.2.: The election considered in Example 7.2 to demonstrate the 2-Copeland$^{0.5}$ and NED rules.

Definition 7.6. Consider an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ with m candidates, a committee size k, and a committee $S = \{c_1, c_2, \ldots, c_k\} \subseteq \mathcal{C}$. For any $\alpha \in [0, 1]$, the *Copeland$^\alpha$ score* of some candidate $c \in S$ is the number of candidates d such that c is preferred to d by a majority of voters plus an α-fraction of the number candidates e such that exactly a half of the voters prefer c to e. The *k-Copeland$^\alpha$ rule* outputs committees consisting of k candidates with the highest Copeland$^\alpha$ scores.

We illustrate the NED and Copeland$^\alpha$ (for $\alpha = 0.5$) rules in Example 7.2.

Example 7.2. Let us again consider an election consisting of four candidates a, b, c, and d and four voters depicted in Figure 7.2, where we also present a table with the NED scores and the 2-Copeland$^{0.5}$ scores of example committees of size two. We show how to compute the score of committee $S = \{a, c\}$; applying the same reasoning allows to compute the NED scores of the other committees. Candidate a is preferred by more than a half of the voters to candidate d and exactly by a half of the voters to candidate b; thus, a contributes two points to the NED score of committee S. Candidate c is preferred exactly by a half of the voters to both b and d. Hence, we have another two points for committee S and we obtain the total score of four.

Next, let us show how to compute the 2-Copeland$^{0.5}$ score of candidate b, which displays a difference between NED and 2-Copeland$^{0.5}$ occurring when there are ties in the numbers of voters preferring one candidate to another. Here, we observe that candidate b is preferred to d by all voters, so b clearly gets one point. Remarkably, the other candidates are preferred to b exactly by a half of the voters. Thus, since we consider the

parameter $\alpha = 0.5$ of 2-Copeland$^\alpha$, b gets a half of a point for each of these comparisons. Eventually, the score of b is two.

Observe that the sets of winning committees under NED and 2-Copeland$^{0.5}$ are different. According to NED committees $\{a, b\}$ and $\{a, c\}$ win the election while according to 2-Copeland$^{0.5}$ only committee $\{a, b\}$ wins the election. Since $\{a, b\}$ and $\{a, c\}$ are the only Gehrlein weakly-stable committees for the considered election, the outcome of 2-Copeland$^{0.5}$ also shows that k-Copeland$^\alpha$ is not Gehrlein weakly-stable.

Detailed studies of Gehrlein stability mostly focused on the weak variant of the notion [BC10]. Some recent findings, as well as results presented in this thesis, suggest that the strong variant is more appealing [Azi+17, SSX17]. For example, all Gehrlein weakly-stable rules are NP-hard to compute [Azi+17], whereas there are strongly-stable rules (such as k-Copeland^0) that are polynomial-time computable. However, we mention that there are approximation algorithms for some Gehrlein weakly-stable rules [SSX17].

7.2.3 Single Transferable Vote (STV)

Multiwinner voting rules, such as STV, can also be round-based. Roughly speaking, in each round, STV picks the most suitable candidate to join the final committee, removes those voters that supported the chosen candidate from further consideration, and repeats this procedure until the committee is selected. In the case of STV, the most suitable candidate is the candidate that is ranked on top most frequently. However, to avoid selecting underrepresented candidates, STV uses the so-called Droop quota [Dro81]. In brief, the Droop quota—being roughly a ratio of the total number of voters and the size of a sought committee–expresses by how many voters a candidate has to be ranked on top to deserve to be elected.

Definition 7.7. Let $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ be an election with m candidates and n voters, k be a committee size, and let $q \coloneqq \lfloor \frac{n}{k+1} \rfloor + 1$ be the Droop quota. The STV rule constructs the winning committee S by executing the following two steps until it picks k candidates to the committee.

1. If there is a candidate c who is ranked first by at least q voters (i.e., whose Plurality score is at least q), then include c in S; remove from the election exactly q voters that rank c first; and remove c from the remaining votes.
2. If such a candidate c does not exist, then remove a candidate d that is ranked first by the fewest voters.

Note that Definition 7.7 does not exactly specify *which* q voters to remove if there are more than q voters that rank first a candidate selected to the committee in some round. Similarly, the definition does not specify *which* candidate to remove if there is more than one that is ranked first by the fewest voters (see Example 7.3 below for an election illustrating this issue). There are couple of ways to cope with these ambiguities. We adopt one that allows to compute all possible winning committees and then collectively outputs them as tied winning committees.

Definition 7.8. A committee S wins under the *parallel-universes (STV)* tie-breaking model if there is any way of breaking ties occurring during applying the STV rule that results in S being elected.

The name of the model from Definition 7.8 comes from an intuitive illustration where the STV rule, whenever it faces a tie, branches out into independent elections, as if happening in parallel universes, according to all possible ways of breaking the tie.

In Example 7.3 we demonstrate how the STV rule works and how to apply the parallel-universes tie-breaking model.

Example 7.3. Consider an election consisting of three candidates a, b, and c and the following voters:

$$\begin{aligned} \text{three voters:} &\quad a \succ b \succ c;\\ \text{two voters:} &\quad c \succ b \succ a;\\ \text{two voters:} &\quad b \succ c \succ a;\\ \text{voter } v^*\text{:} &\quad a \succ c \succ b. \end{aligned}$$

Applying STV to the election, we aim at a committee of size $k := 2$. We first compute the Droop quota, which in our case is $q = \lfloor \frac{8}{3} \rfloor + 1 = 3$. We immediately see that a has the highest Plurality score (which is four) that in turn is greater than q. Thus, we take a to the winning committee and we proceed with removing q voters ranking a at the top position. Observe that if we now remove v^* and remove a from the remaining votes, then the next candidate taken to the winning committee is b, because b has Plurality score three, and we obtain the winning committee of size two. However, if we do not remove v^*, then we need to take c to the winning committee.

> Thus, for both committees $\{a, b\}$ and $\{a, c\}$ there is a way of breaking ties that leads to electing them (one way per committee); as a result, STV with parallel-universes tie-breaking reports these two committees as the winning committees.

We can compute *some* STV winning committee in polynomial time by breaking the internal ties in some arbitrary way, but it is NP-hard to check whether a given committee wins (assuming parallel-universes tie-breaking) [CRX09].

7.2.4 Robustness Radius: Problem Statement

In order to study how robust an election is under some multiwinner rule $\mathcal{M}$, we introduce the following computational problem in which we ask whether a given number of swaps of adjacent candidates is sufficient to change the outcome of the election. Intuitively, the more swaps one needs to change the outcome of an election, the more robust the election is.

> $\mathcal{M}$ Robustness Radius
> **Input:** An election $\mathcal{E}$, a committee size k, and a positive integer ℓ.
> **Question:** Are there ℓ swaps of adjacent candidates in the votes of $\mathcal{E}$ that when applied to election $\mathcal{E}$ give an election $\mathcal{E}'$ such that $\mathcal{M}(\mathcal{E}', k) \neq \mathcal{M}(\mathcal{E}, k)$.

The Robustness Radius problem is strongly related to some other problems studied in the literature. Specifically, in the Destructive Swap Bribery problem (DSB for short) we ask if it is possible to preclude a particular candidate from winning by making a given number of swaps [EFS09, KF19, SYE13]. DSB was already used to study robustness of single-winner election rules by Shiryaev, Yu, and Elkind [SYE13]. We decided to give our problem a different name, and not to refer to it as a multiwinner variant of DSB, because we feel that in the latter the goal should be to preclude a given candidate from being a member of any of the winning committees, instead of changing the outcome in any arbitrary way. In this sense, our problem is very similar to the Margin of Victory problem [BST19, Car11, Mag+11, Xia12], which is also related to the notions of approximation for sublinear winner determination algorithms and sampling of elections [DN15, FT17]; the Margin of Victory problem has the same goal, but instead of counting single swaps, it counts how many votes are changed.

7.3 Classical Computational Complexity

We show that Robustness Radius tends to be computationally challenging. Indeed, we obtain polynomial-time algorithms only for the simplest of our rules: SNTV, k-Approval, and k-Borda.

Theorem 7.1. Robustness Radius *is solvable in polynomial time for SNTV, k-Approval, and k-Borda.*

Proof. Each of our rules proceeds by computing an individual score for each of the candidates (based on this candidate's positions in the preference orders of the voters) and by letting the winning committees consist of the candidates with the highest scores. We first describe a general strategy for dealing with rules of this form and then show how to implement this strategy for SNTV, k-Approval, and k-Borda.

Let $\mathcal{M}$ be one of SNTV, k-Approval, and k-Borda, let $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ be an election, where $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ and $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, and let k be the committee size. For each candidate $c \in \mathcal{C}$, let $\mathrm{score}(c)$ be the individual score of candidate c. Without loss of generality, assume that $\mathrm{score}(c_1) \geq \mathrm{score}(c_2) \geq \cdots \geq \mathrm{score}(c_m)$—we can always achieve this by relabeling the candidates. We are interested in computing a shortest sequence of swaps of adjacent candidates that transforms election $\mathcal{E}$ into some election $\mathcal{E}'$ such that $\mathcal{M}(\mathcal{E}, k) \neq \mathcal{M}(\mathcal{E}', k)$. We consider two disjoint cases:

1. There is a unique winning committee in election $\mathcal{E}$ or
2. there are several tied winning committees in election $\mathcal{E}$.

First, we deal with the case of a unique winning committee $W = \{c_1, c_2, \ldots, c_k\}$. It holds that $\mathrm{score}(c_k) > \mathrm{score}(c_{k+1})$ (otherwise W would not be a unique committee). Consider some arbitrary sequence of swaps that, for the chosen multiwinner voting rule $\mathcal{M}$, transforms $\mathcal{E}$ into some election $\mathcal{E}'$ such that $\mathcal{M}(\mathcal{E}, k) \neq \mathcal{M}(\mathcal{E}', k)$, and consider the first swap whose execution changes the set of winning committees. Prior to this swap, each of the candidates $c_1, c_2, \ldots, c_k$ had its individual score higher than each of the candidates $c_{k+1}, c_{k+2}, \ldots, c_m$. After the swap some candidate from the latter group had its individual score at least as high as one of the members of the former group. Thus, to find a shortest sequence of swaps that changes the result of election $\mathcal{E}$, it suffices to find a shortest sequence of swaps that ensures that some candidate $c \in \mathcal{C} \setminus W$ has a score at least as high as some candidate from committee W. Then, there must be at least two winning committees, one with c and without c.

Now let us consider the case where there are several winning committees. It must hold that $\text{score}(c_k) = \text{score}(c_{k+1})$ and we can partition the set of candidates into three sets, depending on the relation of their score to that of c_k:

$$\mathcal{C}_{\text{above}} \coloneqq \{c_i \subseteq \mathcal{C} \colon \text{score}(c_i) > \text{score}(c_k)\},$$
$$\mathcal{C}_{\text{equal}} \coloneqq \{c_i \subseteq \mathcal{C} \colon \text{score}(c_i) = \text{score}(c_k)\},$$
$$\mathcal{C}_{\text{below}} \coloneqq \{c_i \subseteq \mathcal{C} \colon \text{score}(c_i) < \text{score}(c_k)\}.$$

Each $\mathcal{M}$-winning committee of election $\mathcal{E}$ consists of all candidates from $\mathcal{C}_{\text{above}}$ and an arbitrary subset of $k - |\mathcal{C}_{\text{above}}|$ candidates from $\mathcal{C}_{\text{equal}}$. As in the previous case, let us consider a sequence of swaps that transforms election $\mathcal{E}$ into an election with a different set of winning committees, and consider the first swap after which the set of winning committees changes. The effect of this swap must be that one of the following situations arises:

1. Not all candidates in $\mathcal{C}_{\text{equal}}$ have the same score;

2. all candidates in $\mathcal{C}_{\text{equal}}$ have the same score, but some candidate in $\mathcal{C}_{\text{above}}$ obtains score at most that of the candidates in $\mathcal{C}_{\text{equal}}$; or

3. all candidates in $\mathcal{C}_{\text{equal}}$ have the same score, but some candidate in $\mathcal{C}_{\text{below}}$ obtains score at least that of the candidates in $\mathcal{C}_{\text{equal}}$.

So, to find a shortest sequence of swaps that changes the result of election $\mathcal{E}$, it suffices to find a shortest sequence of swaps that ensures that one given candidate has score higher (or equal) than some other given candidate. For example, to deal with the situation from Item 1, it suffices to try each pair p, d (the names stand for "preferred" and "despised") of distinct candidates from $\mathcal{C}_{\text{equal}}$ and find a shortest sequence of swaps that ensures that p scores better than d does. We treat other possible scenarios listed above analogously.

Let us now stress a consequence of the above reasoning for both the case of a unique winning committee and the case of several winning committees: To prove our theorem it remains to show for each of our three rules a polynomial-time procedure that, given two candidates, p and d, finds a shortest sequence of swaps that ensures that p scores better than (or the same as) d does. We provide such procedures below. Note that we focus on the case of ensuring that p's score is at least that of d. Adapting our reasoning to the case of ensuring that p has a greater score than d is straightforward.

SNTV We guess three nonnegative numbers, B_1, B_2, and B_3. We find B_1 votes where d is ranked first and p is ranked as high as possible, and we shift p to the top position. In effect, d looses its Plurality point and p gains it. Then, we find B_2 votes where p is ranked as high as possible, but not on the first position, and we shift p to the top position. Finally, we find B_3 votes where d is ranked first, and therein we shift d down by one position. If at any point of this algorithm we do not find sufficiently many voters with a given property, then we drop the respective guess of B_1, B_2, and B_3. We check if as a consequence of our swaps p's score is at least the same as that of d and, if so, we store the number of performed swaps. Finally, after considering all possible $O(n^3)$ guesses of B_1, B_2, and B_3, we output the lowest number of swaps observed. Note that our procedure must have succeeded for at least one guess; in particular, for the guess ensuring that all voters rank p on top.

***k*-Approval** We proceed in the same way as in the case of SNTV, but herein the guesses are more intricate. First, we partition the voters into four groups of voters who:

1. neither give a point to p nor to d;
2. give a point to p but not to d;
3. give a point to d but not to p; and
4. give points to both p and d.

Observe that there is no point in affecting the voters in the second group, but there are two ways of altering those in the third group. Hence, we guess numbers B_1, B_3', B_3'', and B_4 of voters whose preference orders we will modify. For the first group, we execute the smallest number of swaps that ensures that B_1 voters give a point to p. For the third group, we execute the smallest number of swaps that ensures that B_3' voters give a point to p and that B_3'' voters do not give a point to d. Note that these operations are, in essence, independent (except for one special case when d is just above p, which can easily be treated separately). For the fourth group, we execute the smallest number of swaps that ensures that B_4 voters do not give a point to d.

***k*-Borda** We perform the following operation until the score of p is at least the same as that of d. We find a vote where p is ranked below d, but the difference between their positions is smallest, and we shift p one position

higher (possibly passing d, if in this vote p is ranked just below d). Note that if the score of p is lower than that of d, then there must be a vote where p is ranked below d, each swap decreases the difference between the scores of p and d by one point or by two points (if p passes d), and our strategy of choosing swaps ensures the highest number of swaps of value two.

This completes the proof. □

We contrast the positive result from Theorem 7.1 for SNTV, k-Approval, and k-Borda with a general NP-hardness of computing the robustness radius for all Gehrlein weakly-stable rules.

Theorem 7.2. Robustness Radius *is* NP*-hard for each Gehrlein weakly-stable rule, even for committee size* $k = 1$.

Proof. We reduce from the NP-hard Exact 3-Set Cover problem [GJ79] where we are given a set $X = \{x_1, x_2, \ldots, x_{3h}\}$ of elements and a set $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of triples of elements of X. We ask whether there are h triples that, together, contain all elements of X. In the following polynomial-time many-one reduction we assume that every element occurs in exactly three input triples; this variant of the problem remains NP-hard [Gon85].

Our reduction proceeds as follows. For each element $x \in X$, we have an *element candidate* $c(x)$. Extending this notation, later in the proof, for a given subset $X' \subseteq X$ of the elements, we write $c(X')$ to denote the set of element candidates that correspond to the members of X' (in particular, $\mathcal{C}(X)$ means the set of all element candidates). We will have $2m + 8h$ voters and for each of them we introduce $4h + 1$ distinct *dummy candidates*. We write $\mathcal{D}$ to denote the set of all these dummy candidates, and for each voter v we write $D(v)$ to denote the set of dummy candidates associated with v. Further, we also have two special candidates p and d. Altogether, we have $2 + 3h + (4h + 1)(2m + 8h)$ distinct candidates, collected in the set

$$\mathcal{C} \coloneqq \{p, d\} \cup c(X) \cup \mathcal{D}.$$

We form the following $2m+8h$ voters, where, in each preference order, the ellipsis represents all candidates not mentioned explicitly, ordered in an arbitrary way (recall that for two disjoint sets of candidates C and D, by writing $C \succ D$, we mean that each candidate in C is before each candidate in D):

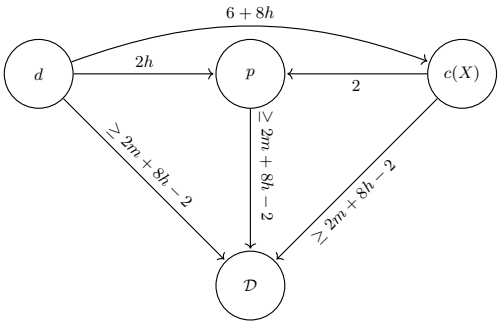


Figure 7.3.: A graph representation of the election constructed by the reduction in the proof of Theorem 7.2. All dummy candidates $\mathcal{D}$ and all element candidates $c(X)$ are contracted to a single vertex. All arcs between vertices forming together one contracted vertex are neglected.

1. For each triple $S \in \mathcal{S}$, there are two voters

$$\begin{aligned} v_S &: \quad d \succ c(S) \succ p \succ D(v_S) \succ \dots \text{ and} \\ \bar{v}_S &: \quad c(X) \setminus c(S) \succ D(\bar{v}_S) \succ p \succ d \succ \dots ; \end{aligned}$$

2. for each $i \in [h-1]$, there is a voter with preference order

$$v_i : \quad d \succ D(v_i) \succ p \succ c(X) \succ \dots ;$$

3. for each $i \in [h+1]$, there is a voter with preference order:

$$v'_i : \quad d \succ c(X) \succ D(v'_i) \succ p \succ \dots ;$$

4. for each $i \in [3h]$, there are two special voters

$$\begin{aligned} v^*_i &: \quad d \succ c(X) \succ D(v^*_i) \succ p \succ \dots \text{and} \\ \bar{v}^*_i &: \quad p \succ d \succ D(\bar{v}^*_i) \succ c(X) \succ \dots . \end{aligned}$$

We form an instance of Robustness Radius that consists of an election with the candidates and voters described above, committee size $k = 1$, and the number B of swaps set to $4h$.

We present the constructed election visually as a weighted graph in Figure 7.3. In this graph, each vertex corresponds either to a single candidate or to a set of candidates. If we have an edge with weight w from a vertex associated with candidate c to a vertex associated with candidate c', then it means that w more voters prefer c to c' than the other way round. For example, there is an arc with weight $6 + 8h$ pointing from candidate d to a vertex associated with $c(X)$. This arc indicates that for every element candidate $c(x)$ (where $x \in X$), the number of voters that prefer d to $c(x)$ is greater by $6 + 8h$ than the number of voters that prefer $c(x)$ to d. To see that this indeed is the case, note that every voter in groups 2, 3, and 4 prefers d to $c(x)$; hence, we have $8h$ voters who prefer d to $c(x)$. In group 1 (consisting of $2m$ voters), d is preferred to x by exactly $m + 3$ voters. Thus, in this group, six more voters prefer $c(x)$ to d than the other way round. The computation is analogous for all other element candidates; thus, candidate d is preferred to each of them by $6 + 8h$ more voters than the other way around.

Now we show that the reduction is correct. Note that, as the committee size is one, if some candidate is a Condorcet winner, then every Gehrlein weakly-stable rule outputs a single winning committee, containing exactly this candidate. Similarly, if there are weak Condorcet winners in the election, then the winning committees are exactly those singletons that contain them. In our election, d is a Condorcet winner (indeed, in Figure 7.3 there are arcs from d to every other vertex), so committee $\{d\}$ wins uniquely.

Let us assume that there is an exact cover of X with h triples from $\mathcal{S}$, and let $I = \{i_1, i_2, \ldots, i_h\}$ be the set of indices of these triples (formally, we have that $\bigcup_{i \in I} S_i = X$). If for each $i \in I$ we shift candidate p to the top of the preference order of voter v_{S_i}, then altogether we make $4h$ swaps and p becomes a weak Condorcet winner. This is so because (i) p is ranked on the fifth place in each of these votes, (ii) the swaps cause p to pass d in h votes (so p ties with d in their head-to-head contest), and (iii) the swaps cause p to pass each element candidate exactly once (so p ties in a head-to-head contest with each element candidate). As a consequence, $\{p\}$ and $\{d\}$ are two winning committees and we see that the election result has changed.

Let us now consider the reverse direction. We first note that if we perform up to $4h$ swaps, then we can change the winning margins indicated in Figure 7.3 by at most $8h$. Thus (assuming that $m \geq 2$), after $4h$ swaps candidate d certainly is still preferred to each candidate other than p by a majority of the voters. Further, after $4h$ swaps still at least half of the voters prefer d to p. From this observation it follows that in each vote either p already is preferred to d or it takes at

least four swaps to move p ahead of d; as a result, with $4h$ swaps, we can make at most h voters prefer p over d and this is just enough to ensure that p and d tie in their head-to-head contest. As a consequence, after $4h$ swaps d certainly is a (weak) Condorcet winner and $\{d\}$ is among the winning committees.

To ensure that $\{d\}$ is not the only winning committee, it is necessary to guarantee that some other candidate is a weak Condorcet winner. Based on Figure 7.3, it is clear that after $4h$ swaps all element candidates and dummy candidates lose at least one head-to-head contest (assuming $m > 1$), so only candidate p may become a weak Condorcet winner. For this to happen, p needs to pass d in h votes and p needs to pass each element candidate in at least one vote. A simple counting argument shows that this is possible only by shifting p to the top position in h votes from the first group that correspond to an exact cover of X with h triples from $\mathcal{S}$.

We conclude by noting that the reduction obviously works in polynomial time. □

Concerning the k-Copeland$^\alpha$ rule, a simple modification of a proof of Kaczmarczyk and Faliszewski [KF19, Theorem 6] (who reduce from Clique) shows that computing the robustness radius for this rule is NP-hard too, even for committees of size $k = 1$. More specifically, to adapt this proof, one needs to only add polynomially-many (with respect to the input) dummy candidates.

Corollary 7.3. *k-Copeland$^\alpha$* Robustness Radius *is* NP*-hard for any $\alpha \in [0, 1]$.*

Without much surprise, we find that Robustness Radius is also NP-hard for β-CC and STV. For these rules, however, the hardness results are, in fact, significantly stronger. In both cases it is already NP-hard to decide whether the outcome of the given election changes after a single swap, and for STV the result holds even for committees of size one. Committee size equal to one cannot yield NP-hardness for β-CC because in this case it is equivalent to the single-winner Borda rule, for which the problem is polynomial-time solvable [SYE13] (this also follows directly from Theorem 7.1).

Theorem 7.4. *β-CC* Robustness Radius *is* NP*-hard and* W[1]*-hard with respect to the size of the committee even if the robustness radius $B = 1$.*

Proof. We show the result by giving a polynomial-time many-one reduction from the Regular Multicolored Independent Set problem. In this problem we are given a regular graph G, where each vertex has degree d and has one of h colors, and we ask if there is an *h-colored independent set*, that is, a

size-h set of pairwise non-adjacent vertices containing one vertex from each color. This problem is known to be both NP-complete and W[1]-hard for the parameter h [Cyg+15, Corollary 13.8]. To obtain our W[1]-hardness result, we will ensure that the reduction uses committees of size that is a function of h only. Specifically, we will use committee size $h + 2$.

Input Instance Let (G, h, d) be an instance of Regular Multicolored Independent Set and let $s \coloneqq |\mathcal{V}(G)|$ and $r \coloneqq |\mathcal{E}(G)|$ be the numbers of, respectively, vertices and edges in the input graph. We assume, without loss of generality, that $s \geq 2h$. Indeed, in a graph with no isolated vertices there is no independent set that contains more than a half of the vertices. Below we describe the election that we use in our β-CC Robustness Radius instance.

Candidates and Committee Size The set of candidates consists of the vertex set $\mathcal{V}(G)$ of graph G, the set $Z \coloneqq \{z_0, z_1, z_2\}$ of *special candidates*, the set $X \coloneqq \{x_1, \ldots, x_h\}$ of *safe candidates*, and the set D of *dummy candidates*. Further in the proof we exactly specify the number of dummy candidates and we show that there are polynomially many of them with respect to $r + s$. We set the committee size $k \coloneqq h + 2$.

High-Level Idea The idea of the construction is to construct our election such that the following holds simultaneously:

1. The *safe committee* $\{z_0, z_1, x_1, \ldots, x_h\}$ always wins (possibly uniquely);
2. for each $\mathcal{V}' \subseteq \mathcal{V}$, if $\mathcal{V}'$ is an h-colored independent set, then $\{z_0, z_2\} \cup \mathcal{V}'$ is a winning committee in the initial election;
3. no other committees apart from those mentioned in the two points above can win;
4. using a single swap of adjacent candidates—which gives the robustness radius of one—it is possible to ensure that the safe committee is the only winning committee. More precisely, a single swap suffices to change the set of winning committees if and only if there is an h-colored independent set for G.

In particular, we will ensure that if there is no h-colored independent set, then the safe committee will have a dissatisfaction (recall Definition 7.1 for the notion

of dissatisfaction) lower by at least four points than the next best committee (so a single swap would not suffice to change the set of winning committees).

Dummy Candidates and Value Δ In our construction we make sure that the safe committee has dissatisfaction

$$\Delta \coloneqq 8r + hs^2,$$

and that this is the lowest possible dissatisfaction (prior to performing swaps). To simplify our construction, we use numerous dummy candidates and we adopt the following convention. Whenever we put a dummy candidate among the top Δ positions in a vote, we put this dummy candidate behind position Δ in all other votes (on its own, this is not enough to guarantee that no dummy candidate belongs to a winning committee, but we will later show that this indeed is the case). As a consequence, for n voters we need at most $O(n\Delta)$ dummy candidates. Since we will form only polynomially many voters, we will also need only polynomially many dummy candidates.

Voters In the following, we partition the voters of our election in four groups, each playing a specific role in the construction. We briefly mention the voters' respective roles and formally prove the correctness of our implementation of the roles later. Whenever we put the symbol " $\ggg$ " in a preference order, we mean listing Δ "fresh" dummy candidates (i.e., ones that are not ranked among the top Δ positions by any other voter), followed by all the remaining candidates in some arbitrary order.

Special Candidate Voters This group consists of $h+3$ voters with preference orders of the form $z_0 \succ \ggg$. These voters ensure that every winning committee includes candidate z_0.

Safe Committee Voters For each color $i \in [h]$, we form $(s+1) \cdot s/2 + 6d$ voters with preference order $x_i \succ z_2 \succ \ggg$. These voters ensure that the safe committee $\{z_0, z_1, x_1, x_2, \ldots, x_h\}$ is indeed winning.

Vertex Selection Voters For each color $i \in [h]$, we add s voters, where each vertex candidate of color i appears exactly once on each of the first s positions, candidate z_1 is ranked on position $(s+1)$, and all other top Δ positions are occupied by the dummy candidates. Formally, we form these votes as follows. We start with s votes with preference orders:

$$
\begin{array}{lllllllllllll}
v_1 & \succ & v_2 & \succ & \cdots & \succ & v_{s-1} & \succ & v_s & \succ & z_1 & \succ & \ggg, \\
v_2 & \succ & v_3 & \succ & \cdots & \succ & v_s & \succ & v_1 & \succ & z_1 & \succ & \ggg, \\
 & \vdots & & & & & & \vdots & & & & & \\
v_s & \succ & v_1 & \succ & \cdots & \succ & v_{s-2} & \succ & v_{s-1} & \succ & z_1 & \succ & \ggg.
\end{array}
$$

Then, we replace each vertex candidate that is not of color i with a fresh dummy candidate. The role of this group is to ensure that except for the safe committee, every other winning committee (if it exists) must contain exactly one vertex of each color.

Independent Set Voters For every edge $\{u, v\}$, we introduce two pairs of voters with preference orders of the forms:

$$u \succ v \succ z_0 \succ \ggg, \text{ and}$$
$$v \succ u \succ z_0 \succ \ggg .$$

The role of this group is to ensure that if there is a winning committee that contains h vertex candidates, then the vertices corresponding to these candidates form an independent set.

This completes the construction. We note that it is executable in polynomial time. Before we formally prove the correctness of our construction, we discuss several important facts about possible winning committees for the constructed election.

Safe Committee In our construction, the safe committee $\{z_0, z_1, x_1, \ldots, x_h\}$ provides a total dissatisfaction equal to Δ. To see this, note that the special candidate voters and the safe committee voters have dissatisfaction zero for the safe committee. For every color, the respective vertex selection voters together have a dissatisfaction equal to s^2. Thus, the dissatisfaction of all vertex selection voters of all colors is hs^2. The independent set voters generate a dissatisfaction equal to $8r$ (for each edge, the two pairs of voters in total have dissatisfaction 8). Altogether, the safe committee has dissatisfaction score $\Delta = 8r + hs^2$.

Independent Set Committees Every committee $\{z_0, z_2\} \cup \mathcal{V}'$, where $\mathcal{V}' \subseteq \mathcal{V}(G)$ is an h-colored independent set, causes total dissatisfaction exactly Δ. Indeed, for such a committee the following holds (we provide additional explanations for the last two voter groups below):

1. Special candidate voters have a dissatisfaction equal to zero;
2. each safe committee voter has a dissatisfaction equal to one (due to candidate z_2), so altogether their dissatisfaction is $h((s+1)\cdot s/2+6d) = (s+1)\cdot hs/2+6hd$;
3. vertex selection voters have total dissatisfaction $h(s-1)\cdot s/2$. To see this, consider a group of vertex selection voters for some color i. As $\mathcal{V}'$ is h-colored, it contains exactly one vertex of color i, which these voters rank on all positions between 1 and s (and they rank all other committee members below these positions). This means that their dissatisfaction is $0+1+\cdots+(s-1) = (s-1)\cdot s/2$. As there are h colors, after multiplying this number by h, we get our total dissatisfaction;
4. independent set voters have total dissatisfaction $8r-6hd$. To show this, we first note that the voters in this group have a dissatisfaction at most $8r$ due to candidate z_0. However, for each edge $\{u,v\}$ such that $\mathcal{V}'$ contains exactly one of the vertex candidates u, v, this dissatisfaction is decreased by six. If our committee contained both u and v, then the dissatisfaction would be decreased by eight, but this does not happen as we assumed $\mathcal{V}'$ to be an independent set. Since our committee contains exactly h vertices and each vertex touches exactly d unique edges (because $\mathcal{V}'$ is an independent set), we have total dissatisfaction $8r-6hd$.

One can verify (and we will show this formally later) that if we replace $\mathcal{V}'$ with a set of h vertices of different colors not forming an independent set, then the dissatisfaction would be higher by at least four points. Briefly, for every two points that we gain by "covering" some edge with two vertices rather than one, we loose six points for being able to cover one edge less.

Losing Committees Next, we show that every other committee causes a total dissatisfaction of at least $\Delta+4$. To this end, we distinguish between five cases for possible committees.

Case 1 (Committees That Do Not Contain z_0) Every committee $\mathcal{C}'$ that does not contain candidate z_0 causes the total dissatisfaction of at least $\Delta+h$. When z_0 is not part of the committee, then up to $k=h+2$ voters from the special candidate voters group have dissatisfaction at least one (in the best case, they are represented by their second-best choice), and the last

one has dissatisfaction at least Δ. Thus, z_0 must belong to all winning committees.

Case 2 (Committees That Contain z_0, z_1, and z_2) Every committee $\mathcal{C}'$ that contains z_0, z_1, and z_2 causes the total dissatisfaction at least $\Delta+4$. To see this, let us first consider the dissatisfaction of the voters when they are represented by $\{z_0, z_1, z_2\}$ only. In this case, the special candidate voters have dissatisfaction zero; the safe committee voters have the dissatisfaction of $h((s+1)\cdot s/2+6d)$; the vertex selection voters have dissatisfaction hs^2; and the independent set voters have dissatisfaction $8r$. Thus, the total dissatisfaction is

$$h\big((s+1)\cdot s/2+6d)\big)+\big(hs^2\big)+\big(8r\big)=\Delta+h((s+1)\cdot s/2+6d.$$

Let us now consider the remaining $h-1$ candidates. Each of the safe candidates can decrease the dissatisfaction by exactly $(s+1)\cdot s/2+6d$. The decrease comes from the vertex selection voters, who, for a given vertex, decrease the dissatisfaction by at most $1+2+\cdots+s$. Each of the vertex candidates can decrease the dissatisfaction by at most $(s+1)\cdot s/2+6d$. This is due to the independent set voters—if an edge is covered by a single vertex candidate, then the dissatisfaction decreases by six; if it is covered by two vertex candidates, then it decreases by eight, but we "split" it over two candidates, so each of them decreases the dissatisfaction by four. We have that $h((s+1)\cdot s/2+6d)-(h-1)((s+1)\cdot s/2+6d)=(s+1)\cdot s/2+6d>4$. That is, altogether the remaining $h-1$ candidates cannot cause the dissatisfaction to be lower than $\Delta+4$.

Case 3 (Committees That Contain z_0 but Not z_2) Consider a committee $\mathcal{C}'$ that contains z_0 and does not contain z_2. If it does not contain all candidates from $\{x_1, x_2, \ldots, x_h\}$, then its dissatisfaction must be (much) larger than 2Δ. For example, if it does not contain some candidate x_i, then at least $(s+1)\cdot s/2+6d-(h+1)>2$ voters with a preference order of the form $x_i \succ z_2 \succ \lll$ are dissatisfied by at least Δ. Thus, let us assume that $\mathcal{C}'$ contains z_0 and all candidates from $\{x_1, x_2, \ldots, x_h\}$. If it does not contain z_1, then—using a similar reasoning as before—the vertex selection voters cause dissatisfaction (much) greater than 2Δ. Summarizing, the safe committee is the only committee that contains z_0, does not contain z_2, and has dissatisfaction lower than $\Delta+4$ (indeed, as we have seen, it has dissatisfaction exactly Δ).

Case 4 (Committees That Contain z_0 but Not z_1) Consider a committee $\mathcal{C}'$ that contains z_0 and does not contain z_1. If this committee does not contain at least a single vertex candidate for each color, then its dissatisfaction is (much) larger than 2Δ. For example, let us assume that $\mathcal{C}'$ does not contain a vertex candidate of color i. Then, $s-(h+1)>1$ of the vertex selection voters corresponding to color i are dissatisfied by at least Δ. Thus let us assume that $\mathcal{C}'$ contains at least one vertex candidate for each color. Then, if $\mathcal{C}'$ does not contain z_2, then it has dissatisfaction (much) greater than 2Δ due to the safe committee voters. In summary, if a committee contains z_0, does not contain z_1, and causes dissatisfaction lower than $\Delta+4$, then it must contain z_2 and a vertex candidate of each color.

Case 5 (Non-Independent Set Committees) Finally, let $\mathcal{C}'$ be a committee of the form $\{z_0,z_2\}\cup\mathcal{V}'$, where $\mathcal{V}'$ contains vertices of each color, but these vertices do not form an independent set. Such a committee causes dissatisfaction at least $\Delta+4$. The special candidate voters have dissatisfaction zero, the safe committee voters have dissatisfaction $h((s+1)\cdot s/2+6d)$, the vertex selection voters have dissatisfaction $h(s-1)\cdot s/2$, and the independent set voters have dissatisfaction at least $8r-6hd+4$. We have analyzed the dissatisfaction of the first three groups of voters when considering the independent set committees; the calculations are the same. Let us, thus, consider the final group of voters. Let q be the number of edges between vertices from $\mathcal{V}'$. There are q edges that are covered twice (i.e., by two vertices from $\mathcal{V}'$), $hd-2q$ edges that are covered once, and all remaining edges are uncovered. The total dissatisfaction of the independent set voters is at least $8r-6(hd-2q)-8q=8r-6hd+4q$. Since $\mathcal{V}'$ is not an independent set, we have $q\geq 1$ and the claim follows.

Correctness of the Reduction The correctness easily follows from the above discussion. On the one hand, if graph G does not contain an h-colored independent set, then the safe committee is the only winning committee with total dissatisfaction Δ and every other committee has dissatisfaction at least $\Delta+4$. Thus, a single swap cannot change the set of winning committees. On the other hand, if graph G does contain an h-colored independent set, then the safe committee is not a unique winning committee. It is easy to verify that then the safe committee does not win anymore if one swaps candidate z_2 with some candidate x_i in some vote from the safe committee group. □

In fact, the proof of Theorem 7.4 implies much more than stated in the theorem. In particular, our construction shows that the problem remains NP-hard even if we are given the current winning committee as a part of the input. Furthermore, the same construction implies that deciding whether a given candidate belongs to some β-CC winning committee is both NP-hard and *co*NP-hard (the NP-hardness result is sometimes taken for granted in the literature, but has not been shown formally yet; see, for instance, Footnote 4 in the work of Bredereck et al. [Bre+16]).

We conclude this section by showing that the Robustness Radius problem is NP-hard for STV even if we consider its single-winner variant (i.e., if we fix the committee size to be one) and consider exactly one swap.

Theorem 7.5. *STV* Robustness Radius *is* NP*-hard even for committee size* $k = 1$ *and robustness radius* $B = 1$.

Proof. We give a polynomial-time many-one reduction from STV Winner Determination, where, given an election, one needs to decide whether a given candidate is an STV winner in the election. This problem is known to be NP-hard [CRX09, Theorem 4] for committee size $k = 1$. Formally, in an instance $\mathcal{I}$ of the STV Winner Determination problem, we are given an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ with n voters, and a distinguished candidate $c \in \mathcal{C}$. We ask if there is a valid run of STV such that c becomes a winner in $\mathcal{E}$. Without loss of generality, we can assume that c is ranked first by some voter.

Based on $\mathcal{I}$, we construct an instance $\mathcal{I}'$ of the STV Robustness Radius problem as follows. We fix the new set of candidates to be $\mathcal{C}' = \mathcal{C} \cup \{d\}$, where d is a dummy candidate needed by our construction. For each voter $v \in \mathcal{V}$, we put d in v's preference ranking right behind c and add two copies of such a modified vote to $\mathcal{I}'$; we call such votes *non-dummy*. Eventually, we obtain an election $\mathcal{E}'$ by adding $2n + 1$ *dummy* voters who rank d first, c second, and then all remaining candidates in some fixed arbitrary order. Candidate d is the unique winner in $\mathcal{E}'$ as it is ranked first by a majority of the voters. If we want to change the outcome of election $\mathcal{E}'$ with a single swap, then we need to swap c and d in the preference order of one of the dummy voters; otherwise d would still be ranked at the top by a majority of the voters. Let us call an election resulting from swapping d and c in the preference of an arbitrary dummy voter as $\mathcal{E}''$.

Observe that if c wins in $\mathcal{I}$, then c also wins in $\mathcal{E}''$. Indeed, STV first eliminates all candidates except for c and d. In such a truncated profile, there

are $2n + 1$ voters who prefer c to d and $2n$ voters who prefer d to c; hence c is the unique winner.

If c is not a winner in $\mathcal{I}$, then c will be eliminated before some other candidate from $\mathcal{C} \cup \{d\}$ in every possible run of STV on $\mathcal{E}''$. Consider some sequence of eliminations performed by STV on $\mathcal{E}''$. At each step in the sequence, either c is eliminated (because it happens to be the candidate ranked at the top by the fewest voters) or there is a candidate x in $\mathcal{C} \setminus \{c\}$ that is ranked first by at least two more non-dummy voters when compared to c. Since in $\mathcal{E}''$, considering only the dummy voters, candidate d is ranked first by exactly $2n$ of them and candidate c is ranked first by exactly 1 of them, we have that overall x must be ranked first by at least one more voter than candidate c. Thus, c will be removed from the election before x and also before d (because candidate x is at the first place of at most $2n - 1$ votes). After c is removed from $\mathcal{E}''$, there will be at least $2n + 1$ voters who rank d first. Thus, d is the unique winner of the election. Consequently, we have shown that the outcome of election $\mathcal{E}'$ can change with a single swap if and only if the answer to the original instance $\mathcal{I}$ is "yes." □

7.4 Parameterized Computational Complexity

We complement our investigations on the complexity of the Robustness Radius problem by showing several algorithms providing fixed-parameter tractability results for a few parameters.

First, we mention that Robustness Radius is fixed-parameter tractable when parameterized by the number of candidates. The proof is implicit, for example, in the works of Dorn and Schlotter [DS12] and Knop, Koutecký, and Mnich [KKM20b], where it is shown that computing the smallest number of swaps leading to ensure a victory of a preferred candidate in a single-winner election is fixed-parameter tractable when parameterized by the number of candidates. The proofs rely on constructing an ILP formulation of the problem where the number of variables is a function of the number of candidates and then applying the result of Lenstra [Len83] from Proposition 2.1. However, to use their technique to solve Robustness Radius, we can easily, without adding new variables, exchange their (set of) inequalities that guarantee that the preferred candidate wins, with a set of inequalities ensuring that a chosen committee does not win. Thus, to find the smallest number of swaps preventing some (originally winning) committee from winning, one can construct an integer linear program for each originally winning committee separately. Analogously,

one can find a series of swaps that guarantees that some committee that was originally not winning is winning after the swaps are executed. Since the number of committees to try is upper-bounded by a function solely of the number of candidates, this approach yields fixed-parameter tractability of Robustness Radius with respect to the number of candidates.

Proposition 7.6. Robustness Radius *for k-Copeland, NED, STV, and β-CC is fixed-parameter tractable when parameterized by the number of candidates.*

The theoretical running time upper bounds achievable using the technique behind Proposition 7.6 (the upper bounds vary slightly depending on to which problem we apply the technique) are of purely classification nature. To the best of our knowledge, so far no experiments has been conducted to investigate the computational efficiency of the approach from Proposition 7.6 in practice. For STV and β-CC we have fixed-parameter tractability not only with respect to the number of candidates, as mentioned above, but also with respect to the number n of voters. For the case of STV, we assume that the committee size k is such that we never need to "delete non-existent voters" and we refer to committee sizes where such deleting is not necessary as *normal*.

Definition 7.9. For an election with n voters, a committee size k is *normal* if and only if $\left(\left\lfloor \frac{n}{k+1} \right\rfloor + 1\right)k \leq n$.

Example 7.4 below illustrates and discusses some cases of non-normal committees. It includes an important case where $k > n$, which yields an assumption we make in our subsequent proofs.

> **Example 7.4.** Clearly, a committee size k is not normal if $k > n$ (where n is the number of voters). Since in each stage of STV (in which we select a candidate to a winning committee) we delete at least $q \geq 1$ voters, we have more voter deletions than voters.
>
> Similarly, taking elections with $n = 12$ voters and committee size $k = 5$ also leads to a non-normal committee. In this case, we would need to delete $q = \lfloor \frac{12}{5+1} \rfloor + 1 = 3$ voters for each committee member, which would require deleting "15 voters out of 12."

Arguably, the cases with non-normal committee sizes are abnormal and indicate that STV should not be used. This is so, because the idea behind STV is that it proportionally represents the voters subsequently transferring

votes "unused" in one round to the next round. A non-normal committee size, however, makes STV use some votes more than once which contradicts this idea. For a more detailed discussion on STV and its multiple variants we refer to a work of Tideman and Richardson [TR00].

Theorem 7.7. *For normal committee sizes, STV* Robustness Radius *is fixed-parameter tractable when parameterized by the number n of voters.*

Proof. Let $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ be the input election with n candidates and let k be the size of the desired committee. Since k is normal, we have that $k \leq n$. For each candidate c, we define the *rank of* c as $\mathrm{rank}(c) \coloneqq \min_{v \in \mathcal{V}}(\mathrm{pos}_v(c))$. Intuitively, the rank of a candidate is the highest position on which the candidate appears in the votes of an election.

First, we prove that a candidate with a rank higher than $k+1$ cannot be a member of a winning committee. For the sake of contradiction, let candidate c with $\mathrm{rank}(c) > k+1$ be a member of some winning committee W. The STV rule adds a candidate to the committee only when the number of voters who rank such a candidate *first* matches or exceeds the quota $\lfloor \frac{n}{k+1} \rfloor + 1$; then, it immediately removes this candidate. Recall that by the assumption on the rank of c, it is never ranked better than on position k. Thus, before c was included in W, STV must have removed some candidate c' from the election without adding it to W (recall that c had to be ranked first by some voter to be included in the committee). Whenever STV eliminates a candidate, it always chooses one with the lowest Plurality score. Since at the moment when c' was removed, the Plurality score of c was equal to zero (because no more than $k < \mathrm{rank}(c)$ candidates had been removed by that moment), we have that the Plurality score of c' also must have been zero. Consequently, removing c' from the election did not affect the top preferences (i.e., candidates ranked the first) of the voters. Hence, right after removing c', STV removed another candidate with zero Plurality score. By repeating this argument sufficiently many times, we conclude that c must have been eventually eliminated, and, so, could not have been added to W. This yields a contradiction and proves our claim.

Second, a direct corollary of the fact that each member of a winning committee has rank at most k is that we can test in FPT-time (with respect to the number of voters) whether a given sequence of swaps has led to changing the result of our election. Indeed, as there are at most kn candidates with rank at most k, the number of winning committees is upper-bounded by $(kn)^k$, which by the assumption that the committee size is normal is at most n^{2n}. Thus, we can

output all winning committees after applying a given sequence of swaps and compare the outcome with the outcome of the original election.

Third, we observe that the robustness radius for our election is at most nk. Indeed, we can take a member of a winning committee and with at most kn swaps we can push it back to have rank $k+1$ or higher. Since a rank greater than k prevents a candidate from belonging to any winning committee, the outcome of the election is changed. For the sake of simplicity (and without weakening any statement of the proof), we again use the assumption that the committee is normal (i.e., $k \leq n$) and we upper-bound nk with n^2. Due to the discussion in this paragraph, from now on, we focus on sequences of at most n^2 swaps.

Fourth, we observe that in order to change the outcome of an election, we should only swap such pairs of candidates that at least one candidate in the pair has rank at most $n^2 + k$. Indeed, consider a candidate c with $\mathrm{rank}(c) > n^2 + k$. After n^2 swaps, the rank of this candidate would still be above k, so it still would not belong to any winning committee (indeed, as without the shifts, the candidate would be eliminated in the initial set of rounds, when the candidates with no first-place votes are eliminated). Thus, a swap of two candidates with ranks higher than $n^2 + k$ cannot affect the set of winning committees (the exact positions of these two candidates have no influence on the STV outcome).

Collecting the bits, recall that we focus on sequences of at most n^2 swaps that involve candidates with ranks at most $n^2 + n$. Note that there are at most $n(n^2 + n)$ candidates with rank at most $n^2 + n$. Thus, there are at most $(2n^3 + 2n^2)^{n^2}$ possible n^2-long sequences, constructed by picking one candidate for each of the n^2 swaps and then choosing whether we swap the candidate with the preceding or with the following candidate. In such a way, we obtain an upper-bound on the number of all swap sequences that we need to check in order to find a shortest one that guarantees a result change. For each sequence of swaps, we test in FPT-time whether the election outcome changes. □

The algorithm for the case of β-CC is more involved. Briefly put, it relies on finding in FPT-time (with respect to the number of voters) either the unique winning committee or two committees tied for victory. In the former case, it combines brute-force search with dynamic programming, and in the latter case, either a single swap or a greedy algorithm suffice. For clarity, we start with presenting the first phase, that is, finding the unique winning committee or two tied committees, as a separate proposition.

Proposition 7.8. *Given an election and a committee size, one can check in* FPT*-time with respect to the number of voters whether the election has a unique*

β-CC winning committee (in which case it outputs this committee) or whether there is more than one β-CC winning committee (in which case it outputs any two winning committees).

Proof. Let $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ be the input election with n voters and let k be the committee size. If $k \geq n$, then every winning committee consists of each voter's most preferred candidate and sufficiently many other candidates to form a committee of size exactly k. Thus, in this case the algorithm can provide the required output in polynomial time, so we assume that $k < n$. To avoid trivial cases, without loss of generality, we also assume that there are more than k candidates.

Our algorithm proceeds by considering all partitions of $\mathcal{V}$ into k disjoint sets (there are at most $k^n \leq n^n$ such partitions). For a partition $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$, the algorithm proceeds as follows (intuitively, the voters in each group $\mathcal{V}_i$ are to be represented by the Borda winner of the election $(\mathcal{C}, \mathcal{V}_i)$):

1. For each election $\mathcal{E}_i = (\mathcal{C}, \mathcal{V}_i)$ we compute, in polynomial time, the set B_i of candidates that are Borda winners of $\mathcal{E}_i$.

2. If each B_i is a singleton and all B_i's are distinct, then we store a single committee $W = B_1, B_2, \ldots, B_k$.

   Otherwise, we form two distinct committees, W_1 and W_2, such that for each B_i, $W_1 \cap B_i \neq \emptyset$ and $W_2 \cap B_i \neq \emptyset$. First, we form a set W_0 by taking the union of all singletons among $B_1, B_2, \ldots, B_k$. Clearly, $|W_0| < k$ because otherwise we would not enter this part of the algorithm. Then, we form a new sequence $B'_1, B'_2, \ldots, B'_t$ of sets by removing from sequence $B_1, B_2, \ldots, B_k$ all those sets that have a nonempty intersection with W_0. If the new sequence turns out to be empty, then every set $\mathcal{V}_i$, where $i \in [k]$, has their Borda winner(s) in W_0. Thus, we form W_1 and W_2 by extending W_0 by adding arbitrary candidates such that W_1 and W_2 are distinct; this is possible because there are more than k candidates in the election. If the new sequence is not empty, then we form W_1 and W_2 as follows. We first include all members of W_0 in both sets. Then, for each B'_i, we include the lexicographically first member of B'_i in W_1 and the lexicographically last one in W_2 (recall that each of the B'_i contains at least two candidates). This ensures that W_1 and W_2 are distinct. If W_1 and W_2 still contain fewer than k candidates, then we extend them by including arbitrary candidates (ensuring that they remain distinct; this is possible because there are more than k candidates in total). After constructing W_1 and W_2, we store both.

We check whether among the stored committees there is a unique committee W such that every other stored committee has a lower β-CC score. If such a committee exists, then we output it as the unique winning committee. Otherwise, there are two stored committees, $W'$and W'', that both have β-CC score not smaller than every other stored committee has. We output W' and W'' as two committees tied for winning. If there is more than one choice for W' and W'', then we pick an arbitrary pair. □

Before we move on to the proof of the fixed-parameter tractability of β-CC Robustness Radius, we introduce some additional notation. Let $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ be some election and let $v \in \mathcal{V}$. By $\mathrm{top}(v)$ we mean the candidate ranked first by v. By $\mathrm{top}(\mathcal{E})$ we mean the set $\{\mathrm{top}(v) \colon v \in \mathcal{V}\}$, that is, the set of candidates who are at least once ranked first in election $\mathcal{E}$. Recall that for a committee W, the representative of some voter v is the member of W that v ranks highest. Finally, for committee W and voter v, we define $\mathrm{reppos}_v(W)$ to be the position of v's representative in W in v's vote.

Theorem 7.9. *β-CC Robustness Radius is fixed-parameter tractable when parameterized by the number of voters.*

Proof. Let $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ be the input election with m candidates and let k be the committee size. Using Proposition 7.8, we check whether there is a unique β-CC winning committee in $\mathcal{E}$ and, depending on the result, we proceed with distinguishing two cases.

There Is a Unique Winning Committee W We first describe a function that encapsulates the effect of shifting forward a particular candidate within a given set of votes. For each voter v, each candidate c, and each nonnegative integer b, we define the vote $\mathrm{shift}(v, c, b)$ obtained from vote v by shifting candidate c by b positions forward, and we define:

$$g(v, c, b) \coloneqq \beta_m(\mathrm{pos}_{\mathrm{shift}(v,c,b)}(c)) - \beta_m(\mathrm{reppos}_{\mathrm{shift}(v,c,b)}(W)).$$

In other words, $g(v, c, b)$ is the difference between the Borda scores of c and the highest-ranked member of W in vote v with c shifted b positions forward.

Let $\mathcal{V}'$ be some subset of voters, and rename the voters so that $\mathcal{V}' = \{v_1, \ldots, v_{n'}\}$. For each candidate c and each nonnegative integer b, we define:

$$g(\mathcal{V}', c, b) \coloneqq \max\left\{ \sum_{i=1}^{n'} g(v_i, c, b_i) \;\middle|\; b_1, \ldots, b_{n'} \geq 0 \text{ and } b_1 + \cdots + b_{n'} = b \right\}.$$

Intuitively, $g(\mathcal{V}', c, b)$ specifies how many points more c would receive from the voters in $\mathcal{V}'$ as their representative than these voters would assign to their representatives from W, if we shifted c by b positions forward in an optimal way.

We assume that $g(\emptyset, c, b) = 0$ for each choice of c and b. We can compute $g(\mathcal{V}', c, b)$ in polynomial time using dynamic programming and the following formula (for each $1 \leq t < n'$):[2]

$$g(\{v_1, \ldots, v_t\}, c, b) = \max_{0 \leq b_t \leq b} g(\{v_1, \ldots, v_{t-1}\}, c, b - b_t) + g(v_t, c, b_t).$$

With the function g in hand, we are ready to describe the algorithm. We consider every partition of $\mathcal{V}$ into k disjoint subsets $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$; fix one such partition. Our goal is to compute the smallest nonnegative integer b such that there is a sequence of nonnegative integers $b_1, \ldots, b_k$ that adds up to b, and a sequence $c_1, \ldots, c_k$ of (not necessarily distinct) candidates so that:

(a) $g(V_1, c_1, b_1) + \cdots + g(V_k, c_k, b_k) \geq 0$ and

(b) there is a committee W' such that $\{c_1, \ldots, c_k\} \subseteq W'$ and $W' \neq W$.

The role of candidates $c_1, \ldots, c_k$ is to be the representatives of the voters from the sets $V_1, \ldots, V_k$, respectively, in a new committee W', distinct from W, that either defeats W or ties with it. More formally, condition (a) ensures that there is a way to perform $b = b_1 + \cdots + b_k$ swaps so that the score of committee W' is at least as large as that of W, and condition (b) requires that $W' \neq W$ and deals with the possibility that candidates in $c_1, \ldots, c_k$ are not distinct.

To compute b, we will need the following function f ($\mathcal{C}'$ is a subset of candidates—we will end up using only polynomially many different ones—$i \in [k]$, and b is a nonnegative integer):

$$f(\mathcal{C}', i, b) \coloneqq \max\left\{ \sum_{j=1}^{i} g(V_j, c_j, b_j) \;\middle|\; c_1, \ldots, c_i \in \mathcal{C}', b_1, \ldots, b_i \geq 0, b_1 + \cdots + b_i = b \right\}.$$

We have that the smallest value of b such that $f(\mathcal{C}', k, b) \geq 0$ is associated with candidates $c_1, \ldots, c_k$ and values $b_1, \ldots, b_k$ that satisfy condition (a) above, under the condition that $c_1, \ldots c_k$ belong to $\mathcal{C}'$. To obtain the smallest value of b that is associated with values $b_1, \ldots, b_k$ and $c_1, \ldots, c_k$ that satisfy both conditions (a) and (b) above, it suffices to compute:

$$b_{V_1, \ldots, V_k} \coloneqq \min\{b \in \mathbb{N} \mid w \in W \wedge f(C - \{w\}, k, b) \geq 0\}.$$

[2] It is possible to compute $g(\mathcal{V}', c, b)$ using a greedy algorithm, but the dynamic programming formulation is far easier and allows us to sidestep many special cases.

The fact that we use sets of the form $\mathcal{C} - \{w\}$ in the invocation of function f ensures that we obtain committees distinct from W. Since we try all $w \in W$, we consider all possible committee different than W. The smallest value $b_{V_1,\dots,V_k}$ over all partitions of $\mathcal{V}$ is the smallest number of swaps necessary to change the outcome of the election.

It remains to show that we can compute function f in polynomial time. This follows by assuming that $f(\mathcal{C}', 0, b) = 0$ (for each $\mathcal{C}'$ and b) and applying the standard dynamic programming technique based on the following formula (which holds for each $i \in [k]$):

$$f(\mathcal{C}', i, b) = \max_{0 \leq b_i \leq b,\, c_i \in \mathcal{C}'} f(\mathcal{C}', i-1, b-b_i) + g(V_i, c_i, b_i).$$

Altogether, the part of the proof where there is a unique β-CC winning committee for $\mathcal{E}$ is complete.

There Are at Least Two Committees That Tie for Victory Let W_A and W_B be two β-CC winning committees for $\mathcal{E}$ that we obtained from invoking the algorithm from Proposition 7.8. We check if there is some voter v whose representatives under W_A and W_B are distinct.

If this is true, then a single swap is sufficient to prevent one of the committees from winning. Let a and b be the (distinct) representatives of v under, respectively, W_A and W_B. Without loss of generality, we assume that a is ranked higher than b; thus $a \notin W_B$ because v does not have a as a representative under W_B. It suffices to swap b with the candidate that precedes b. It is clearly possible since b was ranked below a. It also increases the β-CC score of W_B (since $a \notin W_B$), while the score of W_A either stays the same or decreases (the former happens, for example, if b was ranked just below a and b also belonged to W_A; the latter occurs, for example, if a was the predecessor of b but $b \notin W_A$). In consequence, W_A is not a winning committee after the swap and, thus, the set of winning committees changes.

Let us now consider the case where each voter has the same representative under both W_A and W_B, and let R be the set of the voters' representatives. Hence, $R \subseteq W_A \cap W_B$. Since W_A and W_B are distinct (and, by definition, of the same size), there exist candidates $a \in W_A \setminus W_B$ and $b \in W_B \setminus W_A$; thus, we know that $|R| < k$. We claim that $R = \mathrm{top}(\mathcal{E})$, that is, we claim that each representative is ranked first by some voter. For the sake of contradiction, assume that there is a voter v not represented by its top-preferred candidate. In this case, we can obtain committee W_C by copying W_A and then replacing

candidate a with candidate $\mathrm{top}(v)$. Observe that the representative of voter v is ranked higher under W_C than under W_A. Since all other voters have either the same or even higher-ranked representatives under W_C than under W_A, we get a contradiction to the fact that W_A is a winning committee. Thus our claim holds. As a consequence, all β-CC winning committees for election $\mathcal{E}$ are exactly those that contain all candidates from R. To change the election outcome, we have to transform $\mathcal{E}$ to an election $\mathcal{E}'$ that differ in the top-choice candidates they yield, that is, $\mathrm{top}(\mathcal{E}) \neq \mathrm{top}(\mathcal{E}')$. We consider two types of actions that achieve this effect:

1. Shift some candidate $c \in \mathcal{C} \setminus R$ to the top position of some voter v, thus obtaining election $\mathcal{E}'$ such that $c \in \mathrm{top}(\mathcal{E}')$. By assumption, $c \notin \mathrm{top}(\mathcal{E})$, thus we obtain the requested effect.

2. For some candidate $d \in R$ and each voter v that ranks d on top, shift the highest-ranked member of $R \setminus \{d\}$ to the first position in v'. This creates election $\mathcal{E}'$ such that $\mathrm{top}(\mathcal{E}')$ is strictly contained in $\mathrm{top}(\mathcal{E})$.

Actions of the first type include the cheapest one that creates an election $\mathcal{E}'$ such that $\mathrm{top}(\mathcal{E}') \setminus \mathrm{top}(\mathcal{E}) \neq \emptyset$, whereas actions of the second type include the cheapest one that creates an election $\mathcal{E}'$ such that $\mathrm{top}(\mathcal{E}) \setminus \mathrm{top}(\mathcal{E}') \neq \emptyset$. Thus, it suffices to determine the cheapest action among (polynomially-many) actions of each type and output its cost as the smallest number of swaps necessary to change the outcome of the election. □

It is natural to ask whether Theorem 7.9 holds for other variants of the Chamberlin–Courant rule (i.e., for variants based on scoring functions other than the Borda one). This issue is quite intriguing. While the first part of the proof—where we deal with the case of a unique winning committee—is general and works for any scoring function (indeed, it suffices to replace the Borda scoring function β in the definition of function g with any other scoring rule), the situation of the second part is harder to deal with. Indeed, in the second part of the proof, when we consider the case where not all voters have the same representative, we rely on the fact that a single swap of a representative will increase the score of a committee. This is crucial for our argument, and due to this assumption it does not matter which two specific winning committees W_A and W_B we obtained from Proposition 7.8. Without it, we would have to be more careful in choosing them.

We conclude this section by noting that the Robustness Radius problem for k-Copeland$^\alpha$ and NED is W[1]-hard for the parameterization by the number

of voters. This result is a corollary to a W[1]-hardness proof of Kaczmarczyk and Faliszewski [KF19, Theorem 7] for Copeland$^\alpha$ Destructive Shift Bribery. One can easily adapt this proof by inserting sufficiently many dummy candidates between the non-dummy ones, so that the only reasonable swaps are those that shift the designated candidate backward. Since the proof of Kaczmarczyk and Faliszewski [KF19] uses an odd number of voters, it applies to NED as well, and thus we arrive at Corollary 7.10.

Corollary 7.10. Robustness Radius *for k-Copeland and NED is* W[1]*-hard when parameterized by the number of voters.*

7.5 Beyond the Worst Case: An Experimental Evaluation

In the previous Sections 7.3 and 7.4, we presented a theoretical, worst-case analysis of the computational complexity of Robustness Radius for a number of multiwinner rules. In this section, we go beyond the worst-case analysis and we present results of experiments in which we measure the empirical *average robustness* of our rules—that is, the average number of randomly selected swaps that are necessary to change election results under a particular rule. Note that we indeed do not compute the robustness radius—representing the worst case in the sense that it is the minimum (optimal) number of swaps leading to change of the outcome of an election—but rather try to experimentally assess the average robustness of elections under different voting rules.

In the presented experiments we are not concerned with the running times of our algorithms. Hence, we do not focus on the technical specification of the computing machines we used.

In this section, we excluded from consideration the NED rule. We found finding the average robustness for this rule to be computationally too expensive. However, we expect the results to be similar to the results that we obtained for k-Copeland$^\alpha$.

We performed a series of experiments using five distributions of rankings. Three of them were synthetic ones and the remaining two were based on real-life datasets obtained from the PrefLib [MW13] library of real-life preference data. Regarding the real-life data, we used the dataset of preferences over sushi sets [Kam03] and the dataset with preferences over university courses. We treated the real-life elections as distributions by selecting votes from them uniformly at random. Regarding the synthetic distributions, we used the ones listed below (recall Section 6.2 for the definitions and intuitive descriptions as well as for further literature references):

(i) Impartial Culture (IC),

(ii) Mallows model with parameter ϕ between 0 and 1 and the central order drawn uniformly at random, and

(iii) a mixture of two Mallows models with two separate values of parameters ϕ_1, ϕ_2, and two central orders drawn uniformly and independently at random. Additionally, we draw uniformly at random a value $p \in [0, 1]$ and for each vote that we are to generate, we use the first model with probability p, and the second model with probability $1 - p$.

For each of our five distributions, and for each of the voting rules that we consider, (for k-Copeland$^\alpha$, we took $\alpha = 0.5$) we performed 2000 simulations. In each simulation we had drawn an election containing ten candidates (except for the elections generated from the sushi dataset, where there were only nine candidates) and 30 voters from the given distribution. Then, we repeatedly drew a pair of adjacent candidates uniformly at random and performed a swap until the outcome of the election changed (actually, we never did more than 5000 swaps in order to change the outcome). The average number of swaps required to change the outcome of an election for different rules and for different distributions is depicted in Figure 7.4. We present the results for committee size $k = 3$. We have also performed simulations for $k = 5$ that led to similar conclusions. We note that the standard deviations in our experiments were fairly high (usually close to the value of the reported averages, but sometimes almost twice as large as the value of the reported average). This means that in many elections the required number of random swaps was, in fact, notably smaller than the provided average, and in some elections this number was significantly above the average.

After each randomly chosen swap of adjacent candidates, we ran a standard algorithm (mostly a naïve one; for β-CC we used the ILP formulation by Skowron, Faliszewski, and Lang [SFL16]) for computing the winners of the altered elections under the rule under consideration. However, for STV with parallel-universes tie-breaking, we augmented our approach with observations from Theorem 7.7. Thus, for a committee size k, whenever a swap beyond position $k + 1$ was performed, we withdrew recomputing the winning committees (in the proof of Theorem 7.7 why in the winning committees cannot change as a result of such swaps).

As expected, the average number of swaps required to change an election outcome decreases with the increase of randomness in the voters' preferences.

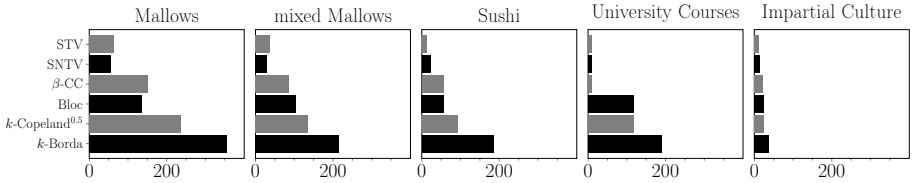


Figure 7.4.: Experimental results showinga the average number of swaps needed to change the outcome of random elections obtained according to the description in Section 7.5. Expectedly, the standard deviations (which are not depicted in the graphs) were quite high, being of the same order as the averages themselves (and often a bit larger).

Indeed, one needs relatively few swaps to change the results of elections generated using the Impartial Culture distribution, but changing the results of elections generated according to the Mallows model requires many more (random) swaps. It is notable that the results regarding the Mallows model are somewhat different from those for the Sushi dataset, as it is often believed that the Mallows model captures the preference orders from the Sushi dataset well [Kam03]. Our results give some circumstantial evidence that there is some nontrivial difference between the Sushi dataset and the Mallows model (which, after all, is to be expected—it is unlikely that a simple synthetic model would capture real-life data perfectly). In particular, based on the fairly small radii of the elections generated using the Sushi distribution, we conclude that the preferences there are rather diverse.

Our analysis shows that among the rules on which we focused, k-Borda, followed by k-Copeland$^{0.5}$, is the most robust. Further, our experiments show that the rules that achieve either diversity (β-CC and, to some extent, SNTV) or proportionality (STV) are usually more vulnerable to small changes in the input. This is aligned with the theoretical insights provided by Bredereck et al. [Bre+21a] (with a minor exception of SNTV); therein the authors show that indeed small changes in the input should have more impact on β-CC, STV than on k-Borda and k-Copeland$^{\alpha}$). For the case of k-Borda, indeed, we would expect that many swaps would cancel each other out (in terms of the effect on the Borda scores of the candidates), which explains the rule's large average robustness. The performance of Borda can also be explained by noting that it is a maximum likelihood estimator for a noise model that is somewhat similar to ours (see, for instance, the overview provided by Elkind and Slinko [ES16]).

The results for STV call for some additional discussion. Indeed, the average robustness of STV turned out to be close to 10 in the Sushi, University Courses, and Impartial Culture distributions, whereas for the Mallows model it was over 60, and for the mixture of two Mallows models it was just below 40. The results for SNTV were qualitatively similar, whereas β-CC typically achieves much higher average robustness values (e.g., in the Sushi dataset its average robustness was more than four times larger than that of STV; for the other datasets—except for the University Courses dataset—it was over two times larger).

From a theoretical perspective, low average robustness of STV is not completely surprising as this rule cannot be easily interpreted as a maximum likelihood estimator [CRX09, CS05] and we should expect lower average robustness from rules focusing on diversity and proportional representation. Yet, the fact that, on average, to change the result of an election with 30 voters and 10 candidates (committee size 3) we may need only about 10 *random* swaps of adjacent candidates is worrisome. In many elections—especially in the low-stake and medium-stake ones—we would expect many voters to make small mistakes, where they rank two adjacent candidates in an opposite order (e.g., because these voters would be tired of the ranking process, or because they would view these two candidates as similar etc.). As a consequence, for small STV elections there is a danger that the outcome is affected by very minor, hard to predict, and hard to observe issues.

Since relatively small STV elections are common in practice (e.g., the rule is used by various universities and their departments for internal elections), this result is quite meaningful. In particular, the organizers of such elections may wish to check if small numbers of random swaps can change the results of their elections and, if so and if this is feasible, they might wish to return to discussions on the voted issues. Of course, this would require some agreement of the voters that if the outcome is not "clear" in the sense of the average robustness, then the discussions are resumed; which, in fact, might be impossible in some setting, but would be quite acceptable in others.

The above discussion is equally applicable to the case of SNTV, but usually when SNTV elections are conducted, the voters only submit their top preferences, so computing the average robustness would be difficult. For the case of β-CC, the test could be executed—and might be meaningful and reasonable—but the danger of non-robust results seems to be smaller than in the case of STV (yet, note that for the University Courses dataset the results of β-CC are as non-robust as those of STV).

7.6 Conclusions

We formalized the notion of robustness of multiwinner rules and studied the complexity of assessing the robustness of collective multiwinner decisions. Our experimental analysis supports the theoretical findings of Bredereck et al. [Bre+21a] and indicates that k-Borda is the most robust among our rules, and that proportional rules, such as STV and the Chamberlin–Courant rule, are on the other end of the spectrum. Indeed, for these rules we suggest that organizers of small-scale elections run tests of the robustness of the obtained results. Notably, for k-Borda, SNTV, and k-Approval, we provided polynomial-time algorithms that can be used to efficiently assess the robustness of an election under these rules.

Our notions of robustness have already attracted attention of other researchers, who have, for example, studied the complexity of the Robustness Radius problem for the Chamberlin–Courant rule in more detail [MS19] (e.g., by considering structured preference profiles) or who have considered the approval setting [GF19, MS19]. Other interesting research directions involve considering counting variants of our problems to assess the probability that a given number of random swaps can change the results (see the initial results of Gawron and Faliszewski [GF19]).

A more open-ended research direction is to seek further notions of robustness, both for the single- and multiwinner voting settings, taking both a practical as well as a theoretical perspective. For example, while in our experiments we focused on random alterations affecting the votes, in our theoretical study of Robustness Radius we sought an *optimal* set of swaps leading to a change in an election outcome. A relevant step to unify these notions would be to theoretically study for how many elections resulting from performing a certain number of swaps in an initial election the outcome under some multiwinner voting rule changes (formally, this question requires studying counting problems). Another collection of open problems would be investigating robustness of elections against a different type of actions than swaps; for example, shifts. Finally, one may want to account for the following facts. First, intuitively performing an action altering the top of a preference order might have more impact that one altering the tail of a preference order. Second, actions at the top might be rarer as people tend to be more sure about their preferred choices. So, one needs to investigate scenarios which take into account where in a preference order a particular action occurs. Such cases seem to provide a range of open problems concerning both theoretical and experimental studies.

The Robustness Radius problem models a somehow extreme situation where the goal is to change the result of an election in any way, without a clear goal. In Chapter 8, we study a scenario where a group of manipulators coordinate to obtain a specific, desired outcome. Observe that this scenario could provide another measure of robustness—the more robust an election is, the higher number of manipulators is needed to change the outcome. Even though we do not directly study this measure in Chapter 8, we provide algorithmic tools (and settle computational lower bounds) which could be of use for further studies on different notions of robustness.

CHAPTER 8

Coalitional Manipulation for Multiwinner Elections

In this chapter, we provide the first in-depth study of the computational complexity of coalitional strategic voting in multiwinner elections. We focus on shortlisting of candidates—that is, selecting a group of "best" candidates—as a special case of multiwinner elections. Specifically, we analyze the perhaps most basic voting rule in this scenario, t-Approval (every voter approves t candidates). In particular, we investigate the influence of several different group evaluation functions (e.g., egalitarian versus utilitarian) and tie-breaking mechanisms modeling pessimistic and optimistic manipulators. Among other things, we conclude that whereas in the utilitarian variant strategic voting is computationally tractable, in the egalitarian setting strategic voting may be computationally intractable regardless of the tie-breaking rule. Altogether, we provide a fairly comprehensive picture of the computational complexity landscape of this scenario. We also conduct preliminary experiments providing strong evidence that the computational intractability of the egalitarian variant can be overcome in practice by fixed-parameter tractable algorithms that we present.

8.1 Introduction

Assume that a university wants to select the two favorite pieces in classical style to be played during the next graduation ceremony. The students were asked to submit their favorite pieces. Then a jury consisting of seven members (three juniors and four seniors) from the university staff selects from the six most frequently submitted pieces as follows: Each jury member approves two pieces and the two winners are those obtaining most of the approvals. The six options provided by the students are "Beethoven: Piano Concerto No. 5" (b_1), "Beethoven: Symphony No. 6" (b_2), "Mozart: Clarinet Concerto" (m_1), "Mozart: Jeunehomme Piano Concerto" (m_2), "Uematsu: Final Fantasy" (o_1), and "Badelt: Pirates of the Caribbean" (o_2). The three junior jury members are excited about recent audio-visual presentation arts (both interactive and passive) and approve o_1 and o_2. Two of the senior jury members are Mozart enthusiasts, and the

other two senior jury members are Beethoven enthusiasts. Hence, when voting truthfully, two of them would approve the two Mozart pieces and the other two would approve the two Beethoven pieces. The winners of the selection process would be o_1 and o_2, both receiving three approvals whereas every other piece receives only two approvals.

The senior jury members meet every Friday evening and discuss important academic issues including the graduation ceremony music selection processes, why "movie background noise" recently counts as classical music [Glo20], and the influence of video games on the ability of making important decisions. During such a meeting they agreed that a graduation ceremony should always be accompanied by pieces of traditional, first-class composers. Thus, finally all four senior jury members decide to approve b_1 and m_1, so these two pieces are played during the graduation ceremony.

This toy example illustrates important aspects of strategic voting in multiwinner elections. In case of coalitional manipulation for single-winner elections (where a coalition of voters casts untruthful votes in order to influence the outcome of an election; a topic which has been intensively studied in the literature [BR15, CW16]) one can always assume that a coalition of manipulators agrees on trying to make a distinguished alternative win the election. In case of *multiwinner elections*, however, already determining concrete possible goals of a coalition seems to be a demanding task: There may be exponentially many (in the input size) different outcomes which can be reached through strategic votes of the coalition members and each member could have its individual evaluation of these outcomes.

Multiwinner voting rules come up very naturally when one has to select from a large set of candidates a smaller set of "the best" candidates. For this selection, various criteria, such as proportional representation, diversity, or excellence [Elk+17], can be interesting. We focus on the last scenario. Here, the goal is to select the best (say highest-scoring) group of candidates. Aiming at excellence comes very naturally in the context of shortlisting, where the objective is to find a *short list* of candidates selected from an initial, much larger list of candidates. For instance, a human resource department wanting to fill a vacancy would select, from all job candidates, a short list of prospective applicants who should be further assessed to find the best fitting applicant [Ide20]. This example neatly illustrates the universal purpose of shortlisting, that is, saving effort at the same time increasing the quality of evaluating suitable candidates. Indeed, human resource departments will either waste a lot of time and effort interviewing every applicant in detail or they will significantly decrease the

quality of interviewing to speed up the process unless they apply shortlisting beforehand.

A standard way of candidate selection in the context of shortlisting is to use scoring-based voting rules. We focus on the two most natural ones: SNTV (single non-transferable vote—each voter gives one point to one candidate) and t-Approval (each voter gives one point to each of t different candidates, so SNTV is the same as 1-Approval).[1] Obviously, for such voting rules it is trivial to determine the score of each individual candidate.

The main goal of our work is to model and understand coalitional manipulation in a computational sense—that is, to introduce a formal description of how a group of manipulators can influence the election outcome by casting strategic votes and whether it is possible to find an effective strategy for the manipulators to change the outcome in some desirable way. We find studying coalitional manipulability from the computational complexity point of view relevant for two main reasons. First, in a natural way we complement well-known work on manipulation for single-winner rules initiated by Bartholdi III, Tovey, and Trick [BTT89], coalitional manipulation for single-winner rules initiated by Conitzer, Sandholm, and Lang [CSL07], and (non-coalitional) manipulation for multiwinner rules initiated by Meir et al. [Mei+08]. Second, we provide efficient algorithms that allow for experimental study of coalitional manipulation that might be interesting both for verifying how likely is or what is an impact of coalitional manipulation in practice (analogously to studies for the single-winner case [BNW11, CT07, Dav+14, Erd+15, Lu+12, Wal11]) and for interdisciplinary study on human behavior when manipulating (like the one recently conducted for multiwinner elections by Scheuerman et al. [Sch+19]).

In coalitional manipulation scenarios, given full knowledge about other voters' preferences, one has a set of manipulative voters who want to influence the election outcome in a favorable way by casting their votes strategically. To come up with a useful framework for coalitional manipulation for multiwinner elections, we first have to identify the exact mathematical model and questions to be asked. A couple of straightforward extensions of coalitional manipulation for single-winner elections or (non-coalitional) manipulation for multiwinner elections do not fit. Directly extending the single-winner variant, one would probably assume that the coalition agrees on making a distinguished candidate part of the winners or that the coalition agrees on making a distinguished

[1] Although some experts argue that t-Approval is not a proper rule for shortlisting applications [BC08, Elk+17], this rule seems quite frequent in practice—for example, "The Board of Research Excellence" in Poland was elected using a variant of t-Approval [Min19].

candidate group part of the winners. The former is unrealistic because in multiwinner settings one typically cares about more than just one candidate—especially in shortlisting it is natural that one wants rather some group of "similarly good" candidates to be winning instead of only one representative of such a group. The latter—that is, agreeing on a distinguished candidate group to be part of the winners—is also problematic since there may be exponentially many "equally good" candidate groups for the coalition. Notably, this was not a problem in the single-winner case; there, one can test for a successful manipulation towards each possible candidate avoiding an exponential increase of the running time (compared to the running time of such a test for a single candidate).

We address the aforementioned issue of modeling coalitional manipulation for multiwinner elections by extending a single-manipulator model for multiwinner rules of Meir et al. [Mei+08]. In their work, the manipulator specifies the utility of each candidate and the utility for a candidate group is obtained by adding up the utilities of each group member. We build up on their idea and let each manipulator report the utility of each candidate. However, aggregating utilities for a coalition of manipulators (in other words, computing a single utility of coalition by aggregating the utilities of manipulators) becomes conceptually demanding—this is especially true for a coalition of manipulators who have diverse utility values for single candidates but still have strong incentives to work together (e.g., as we illustrated in our introductory example).

We only consider coalitions that are fixed, that is, irrespectively of how different the opinions of manipulators are, none of them leaves the coalition. At first glance, this assumption might look too restrictive and unrealistic. However, we believe there are good reasons for making this assumption. First, changing coalitions in the real world usually requires a significant overhead (e.g., formal agreements and negotiations that cost both money and time) which makes such a change rather a last resort. This holds true especially if a coalition is aiming at long-term benefits as, for example, strategic cooperation among firms or governments. Second, there are real-world cases where coalitions are forced, for example, in hierarchical administrative divisions (and their local governments) in countries. Third, computing a best possible manipulation for a given coalition is an important step in deciding whether it is useful to attempt to form such a coalition. To wrap up, instead of focusing on coalition dynamics, we rather concentrate on an analysis of a strength of, intuitively speaking, potential, fixed, or forced coalitions.

Our Contributions We devise a formal description of coalitional manipulation in multiwinner elections arriving at a new, nontrivial model capturing two types of manipulators' attitudes and a few natural ways of utility aggregation. To this end, in our model, we distinguish between optimistic and pessimistic manipulators and we formalize aggregation of utilities in a utilitarian and an egalitarian way.

Using our model, we analyze the computational complexity of finding a successful manipulation for a coalition of voters, assuming elections under rules from the family of t-Approval voting rules. We show that, even for these fairly simple rules, the picture of the computational complexity of coalitional manipulation is diverse. In particular, we observe that finding a manipulation that maximizes the utility of a worst-off manipulator (egalitarian aggregation) is NP-hard (regardless of the manipulators' attitude). This result stands in sharp contrast to the polynomial-time algorithms that we give for finding a manipulation maximizing the sum of manipulators' utilities (utilitarian aggregation). Additionally, we show how to circumvent the NP-hardness for the egalitarian aggregation providing an (FPT) algorithm that is efficient for scenarios with few manipulators and few different utility values that manipulators assign to agents. We survey our computational complexity results in Table 8.1 (Section 8.6).

Related Work To the best of our knowledge, there is no previous work on the computational complexity of *coalitional* manipulation in the context of multiwinner elections. We refer to recent textbooks for an overview on the huge literature on single-winner (coalitional) manipulation [BR15, CW16]. Most relevant to our work, Lin [Lin11] showed that coalitional manipulation in single-winner elections under t-Approval is solvable in linear time by a greedy algorithm. Meir et al. [Mei+08] introduced (non-coalitional) manipulation for multiwinner elections. While pinpointing manipulation for several voting rules as NP-hard, they showed that manipulation remains polynomial-time solvable for k-Approval—a rule that can be interpreted as a multiwinner equivalent of 1-Approval. Obraztsova, Zick, and Elkind [OZE13] further extended the latter result for different tie-breaking strategies and identified additional tractable special cases of multiwinner scoring rules. Yet, they conjectured manipulation to be hard in general for (other) scoring rules. Summarizing, t-Approval is simple but comparably well-studied and as such it is very suitable for serving as a showcase for our study of the presumably computationally harder *coalitional* manipulation.

Organization Section 8.2 introduces basic notation and formal concepts. It also describes our model for coalitional manipulation in multiwinner elections, its variants with respect to different ways of evaluating candidate groups (utilitarian vs. egalitarian), and two kinds of manipulators behavior (optimistic vs. pessimistic). In Section 8.3, we present algorithms and complexity results for computing the output of several tie-breaking rules that allow to model optimistic and pessimistic manipulators. In Section 8.4, we formally define the coalitional manipulation problem and explore its computational complexity using t-Approval as a showcase. Later, in Section 8.5 we present the preliminary results demonstrating practical applicability of the algorithms we develop. We refer to our conclusion in Section 8.6 and Table 8.1 therein for a detailed overview of our findings.

8.2 Preliminaries

In this chapter, following Debord [Deb92], we use the name *k-excellence-group*, abbreviated to *k-egroup*, for committees of size k selected by multiwinner voting rules.[2] Thus we emphasize our focus on shortlisting and bring our terminology closer to shortlisting (real-life) applications where the word "committee" traditionally rather refers to voters and not to candidates (especially frequently in the term "selection committee"). For the sake of brevity, we use *egroup* if the size of an excellence-group is either not relevant or clear from the context.

In the following sections, we formally define and explain our model and the respective variants, which we also motivate with short real-world examples. To this end, we discuss how we evaluate an egroup in terms of utility for a coalition of manipulators and introduce tie-breaking rules that model optimistic or pessimistic viewpoints of the manipulators.

8.2.1 Evaluating Excellence-Groups

As already discussed in the introduction, one should not extend the model of coalitional manipulation for single-winner elections to multiwinner elections in the simplest way (e.g., by assuming that the manipulators agree on some distinguished candidate or on some distinguished egroup) as it would badly harm the expressiveness of coalitional manipulation. Instead, we follow Meir et al. [Mei+08] and assume that we are given a utility function evaluating the candidates for each manipulator and a minimum utility of an egroup, that, if achieved, indicates a successful manipulation.

[2] We modified Debord's term "elite" as we feel that it might carry negative connotations.

Considering a collection of such utility functions there are several ways, each coming with distinct features, of computing the utility of an egroup. In this chapter, we study the following three variants: *utilitarian*, *egalitarian*, and *candidate-wise egalitarian*.

In the *utilitarian variant* (considered by Meir et al. [Mei+08]) the utility of an egroup is the sum of utility values assigned by each manipulator to every candidate in the egroup. This is perhaps the most intuitive way of evaluating the utility of an egroup. Although it does not provide any guarantee on a single manipulator's utility after a manipulation (it might even happen that some manipulator is significantly worse off compared to voting sincerely as illustrated in Example 8.1), the utilitarian variant is justified if the manipulators are able to "internally" compensate such losses, for example, by paying money to each other. For a real-world example, imagine an international company with branches (voters) scattered around the world considering actions to be taken (candidates) in order to reduce its carbon footprint. Seeking the most efficient solution, the company surveys the branches for pointing the actions that reduce the footprint most (utilities). Imagine there is a country that subsidizes companies that reduce the emissions the most. It is natural for all branches in this country to coordinate and vote strategically to force implementation of the most effective footprint-reducing actions. Moreover, it is understandable that the office branches, whose footprint is rather small and thus so is the possible reduction, will rather support the actions that help the factory branches to reduce the footprint. To compensate, the company might decide to distribute more money from the received benefit to the office branches.

Example 8.1. Consider the election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ where $\mathcal{C} = \{b_1, b_2, m_1, m_2, o_1, o_2\}$ is a set of candidates and $\mathcal{V} = \{v_1, v_2, v_3\}$ is the following multiset of three votes:

$$v_1, v_2 \colon \quad o_1 \succ o_2 \succ m_1 \succ m_2 \succ b_1 \succ b_2,$$
$$v_3 \colon \quad m_2 \succ m_1 \succ b_2 \succ b_1 \succ o_1 \succ o_2.$$

Additionally, consider two manipulators, u_1 and u_2, that report utilities to the candidates as depicted in the table below.

$u(\cdot)$	b_1	b_2	m_1	m_2	o_1	o_2
u_1	10	5	4	0	0	0
u_2	1	2	5	7	0	0

Let us analyze the winning 2-egroup under the SNTV voting rule. Observe

> that if the manipulators vote sincerely, then together they give one point to b_1 and one to m_2 (one point from each manipulator). Combining the manipulators' votes with the non-manipulative ones, the winning 2-egroup consists of candidates o_1 and m_2 that both have score two; all other candidates have lower score, so tie-breaking is unnecessary. The utility of egroup $\{m_1, o_1\}$ is equal to seven (with respect to the utilitarian evaluation). Manipulator u_2's utility is seven. However, both manipulators can do better by giving their points to candidate b_1. Then, the winners are candidates o_1 and b_1, giving the total utility of 11 (according to the utilitarian variant). Observe that in spite of the growth of the total utility, the utility value gained by u_2, which is one, is lower than in the case of sincere voting.

The *egalitarian variant* comes in handy, for the scenarios where it is essential to guarantee a certain level of utility for every manipulator. Specifically, the utility of an egroup is the utility of a manipulator whose sum of utilities of candidates from the egroup is the smallest; thus, the egalitarian variant aims at maximizing this number. For a real-world example, imagine a parliament (voters) deciding about possible steps to reduce particulate matter pollution (candidates). Seeking a way to reduce particulate matter below a certain threshold, a coalition of representatives from districts currently not meeting the threshold decides to vote strategically. Naturally, particulate matter reduction (utilities) are differently affected by different steps in the respective districts. The goal of the coalition is that even the worst district is below the threshold, which corresponds to egalitarian aggregation.

The *candidate-wise egalitarian variant* models again scenarios where, as in the utilitarian variant, the overall utilities from members of an egroup are summed up. The utility of the respective candidates, however, is aggregated in a pessimistic way, that is, assuming the lowest utility assigned by any member of the coalition is taken into the sum. For a real-world example, imagine a parliament (voters) deciding on different actions (candidates) to support the economy after a crisis. Representatives of the same party naturally work together as a coalition of strategic voters. Each representative has a different prediction from their own group of experts on the effectiveness of the actions (utilities), which at the end will sum up. To be on the safe side, the coalition decides to take into account for their decision the most pessimistic evaluation any expert group makes for a respective candidate, which corresponds to our candidate-wise egalitarian aggregation variant.

We formalize the described variants of k-egroup evaluation (for r manipulators) in Definition 8.1.

Definition 8.1. Let $\mathcal{C}$ be a set of candidates, $S \subseteq \mathcal{C}$ be an egroup, $U = \{u_1, u_2, \ldots, u_r\}$ be a family of manipulator utility functions where $u_i \colon \mathcal{C} \to \mathbb{N}$, $i \in [r]$. Then the utility of S is:

- $\text{util}_U(S) \coloneqq \sum_{u \in U} \sum_{c \in S} u(c)$ in the *utilitarian variant*,
- $\text{egal}_U(S) \coloneqq \min_{u \in U} \sum_{c \in S} u(c)$ in the *egalitarian variant*, and
- $\text{candegal}_U(S) \coloneqq \sum_{c \in S} \min_{u \in U} u(c)$ in the *candidate-wise egalitarian variant*.

Intuitively, these functions determine the utility of a k-egroup S according to, respectively, the utilitarian and the egalitarian variants of evaluating S by a group of r manipulators (identifying manipulators with their utility functions). We omit subscript U when U is clear from the context. To illustrate Definition 8.1 we apply it in Example 8.2.

Example 8.2. Consider a set $\mathcal{C} = \{b_1, b_2, m_1, m_2\}$ of candidates and two manipulators u_1, u_2 whose utility functions over the candidates are depicted in the table below.

$u(\cdot)$	b_1	b_2	m_1	m_2
u_1	10	5	4	0
u_2	1	2	5	7

Then, evaluating the utility of 2-egroup $S = \{b_1, m_1\}$ by applying the three different evaluation variants gives:

- $\text{util}(S) = (10 + 4) + (1 + 5) = 20$,
- $\text{egal}(S) = \min\{(10 + 4); (1 + 5)\} = 6$, and
- $\text{candegal}(S) = \min\{10, 1\} + \min\{4, 5\} = 5$.

Observe that whereas $\{b_1, m_1\}$ is the optimal k-egroup for the utilitarian variant, it is not the case for the egalitarian variant according to which $\{b_1, m_2\}$ obtains the maximum utility 8. For a change, the optimal choice of k-egroup for the candidate-wise egalitarian variant is $\{b_2, m_1\}$ yielding utility 6.

Analyzing Example 8.2, we observe that we can compute the utilitarian value of egroup S by summing up the overall utilities that each candidate in S contributes to all manipulators; for instance candidate b_1 always contributes the utility of $11 = 10 + 1$ to the manipulators, independently of other candidates in the egroup. Following this observation, instead of coping with a collection of utility function, we can "contract" all manipulator functions to a single function. The new function assigns each candidate a utility value equal to the sum of utilities that the contracted functions assign to this candidate. Analogously, we can deal with the candidate-wise egalitarian variant by taking the minimum utility associated to each candidate as the utility of this candidate in a new function. Thus, in both variants, we can consider a single utility function instead of a family of functions. This is more formally presented in the following observation.

Observation 8.1. *Let $\mathcal{C}$ be a set of candidates, $U = \{u_1, u_2, \dots, u_r\}$ be a family of utility functions such that $u_i \colon \mathcal{C} \to \mathbb{N}$, $i \in [r]$, and* $\mathrm{eval} \in \{\mathrm{util}, \mathrm{candegal}\}$ *be an evaluation function. Then, there always exists a (single) utility function $u' \colon \mathcal{C} \to \mathbb{N}$ such that for every egroup $S \subseteq C$ it holds that* $\mathrm{eval}_U(S) = \sum_{c \in S} u'(c)$.

Proof. We fix some (non-empty) set $\mathcal{C}$ of candidates. Consider a multiset of manipulator utility functions $U = \{u_1, u_2, \dots, u_r\}$ and an egroup $S \subseteq \mathcal{C}$. For the utilitarian variant, create a new utility function u' that assigns to each candidate the sum of utilities given to this candidate by all manipulators; that is, $u'(c) \coloneqq \sum_{i \in [r]} u_r(c)$ for all $c \in \mathcal{C}$. For each candidate function u' returns the sum of utilities given to a candidate by all functions from family U, so

$$\mathrm{util}_U(S) = \sum_{i \in [r]} \sum_{c \in S} u_i(c) = \sum_{c \in S} \sum_{i \in [r]} u_i(c) = \sum_{c \in S} u'_i(c).$$

We follow a similar strategy proving Observation 8.1 for the candidate-wise egalitarian evaluation. We introduce a function u' defined as $u'(c) \coloneqq \min_{u \in U} u(c)$ for each candidate $c \in \mathcal{C}$. Naturally,

$$\mathrm{candegal}_U(S) = \sum_{c \in S} \min_{u \in U} u(c) = \sum_{c \in S} u'(c).$$

□

8.2.2 Breaking Ties

According to Definition 6.2, a multiwinner voting rule (for a committee size k) returns a set of tied k-egroups; hence, to select a single k-egroup from the set of co-winning k-egroups one has to consider tie-breaking rules.

Definition 8.2. A *multiwinner tie-breaking rule* is a mapping that, given an election and a family of co-winning k-egroups, outputs a single k-egroup.

Among different tie-breaking rules, there is a collection of natural rules that is of particular interest in order to model the behavior of manipulative voters. Indeed, in addition to simple lexicographic and randomized tie-breaking rules, both *pessimistic* and *optimistic* tie-breaking rules have already been used to model the manipulator's behavior in case of a single manipulator [Mei+08, OZE13]. To model optimistic and pessimistic manipulators in a meaningful manner we follow this path and make use of the model introduced by Obraztsova, Zick, and Elkind [OZE13]. Here, a manipulative voter v is described not only by its preference order $\succ$ over the candidates but also by a utility function $u\colon \mathcal{C} \to \mathbb{N}$, which allows to express preferences over candidates more precisely than an order. Indeed, we cannot simply use ordinal preferences as it is insufficient to use the fixed lexicographic order of the manipulators' preferences (resp. the reverse of it) over candidates to model optimistic (resp. pessimistic) tie-breaking already in case of a single manipulator [OZE13]. For example, it is a strong restriction to assume that a manipulator would always prefer its first choice together with its fourth choice towards its second choice together with its third choice. It might be that only its first choice is really acceptable (in which case the assumption is reasonable) or that the first three choices are comparatively good but the fourth choice is absolutely unacceptable (in which case the assumption is wrong). To cover this in the tie-breaking process, *coalition-specific* tie-breaking rules get—in addition to the original election, the manipulators' votes, and the co-winning excellence-groups—the manipulators' utility functions in the input. We discuss the formal implementations of these rules and their properties in the remainder of this section.

We start by briefly discussing some necessary notation and central concepts that will allow us to tailor Definition 8.2 to t-Approval, our special case of interest.

Definition 8.3. Let $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ be an election, k be the size of the egroup to be chosen, and $\mathcal{M}$ be a scoring-based multiwinner voting rule. Then, the set $\mathcal{C}^+ \coloneqq \bigcap_{S \in \mathcal{M}(\mathcal{E},k)} S$ of *confirmed candidates* contains the candidates that are in all co-winning k-egroups; the set $\mathcal{P} \coloneqq \bigcup_{S \in \mathcal{M}(\mathcal{E},k)} S \setminus \mathcal{C}^+$ of *pending candidates* contains the candidates that are in some co-winning committee but not in all of them; and the set $\mathcal{C}^- \coloneqq \mathcal{C} \setminus (\mathcal{C}^+ \cup \mathcal{P})$ of *rejected candidates* consists of all candidates that are in no co-winning committee.

Naturally, $|\mathcal{C}^+| \leq k$, $|\mathcal{C}^+ \cup \mathcal{P}| \geq k$, and (for some fixed scoring-based multiwinner voting rule) every candidate from $\mathcal{P} \cup \mathcal{C}^-$ receives fewer points than every candidate from $\mathcal{C}^+$. Additionally, all pending candidates receive the same number of points.

We define the following families of tie-breaking rules which are considered in this chapter. In order to define optimistic and pessimistic rules, we assume that in addition to $\mathcal{C}^+$, $\mathcal{P}$, and k, we are given a family of utility functions. The given data is sufficient to evaluate the k-egroups as discussed in Section 8.2.1 even though it does not contain full information about an election. To emphasize that election data is unnecessary, in Definition 8.4 we introduce the name "tie-breaking perspective" to denote the input of tie-breaking rules.

Definition 8.4. For a set $\mathcal{C}$, a *tie-breaking perspective* $(\mathcal{C}^+, \mathcal{P}, k, U)$ *over* $\mathcal{C} \supseteq \mathcal{C}^+ \cup \mathcal{P}$ is a quadruple consisting of disjoint sets $\mathcal{C}^+$ of confirmed candidates and $\mathcal{P}$ of pending candidates, a committee size k such that $\left|\mathcal{C}^+\right| \leq k < \left|\mathcal{C}^+\right| + |\mathcal{P}|$, and a family U of utility functions over candidates in $\mathcal{C}$.

Tie-breaking perspectives form the input of all multiwinner tie-breaking rules we study in this chapter (formally, lexicographic tie-breaking does not require any family of utility function but we keep it in the input of this tie-breaking rule for consistency). For the sake of defining further concepts using tie-breaking perspectives, we usually do not need to exactly know all elements of set $\mathcal{C}$ besides the candidates in $\mathcal{C}^+$ and $\mathcal{P}$. Thus, in the following Definition 8.5, and generally whenever it does not lead to an error or ambiguity, we omit the part "over $\mathcal{C}$." Note, however, that if we wanted to compute the outcome of a tie-breaking rule for a given tie-breaking perspective, we have to know the whole set $\mathcal{C}$ since the tie-breaking rule might depend on the utility functions of candidates other than those in $\mathcal{C}^+$ and $\mathcal{P}$.

Definition 8.5. Let $(\mathcal{C}^+, \mathcal{P}, k, U)$ be a tie-breaking perspective, $\mathcal{F}$ be a multiwinner tie-breaking rule, and $\mathrm{eval} \in \{\mathrm{util}, \mathrm{egal}, \mathrm{candegal}\}$. Then:

- Tie-breaking rule $\mathcal{F}$ belongs to the family $\mathcal{F}_{\mathrm{lex}}$ of *lexicographic tie-breaking rules* if and only if ties are broken lexicographically with respect to some predefined order $>_{\mathrm{lex}}$ of the candidates from $\mathcal{C}^+ \cap \mathcal{P}$. That is, $\mathcal{F}$ outputs a committee consisting of all candidates from $\mathcal{C}^+$ and the top $k - |\mathcal{C}^+|$ candidates from $\mathcal{P}$ with respect to $>_{\mathrm{lex}}$.

- Tie-breaking rule $\mathcal{F}$ belongs to the family $\mathcal{F}_{\mathrm{opt}}^{\mathrm{eval}}$ of *optimistic tie-breaking rules* if and only if it always outputs some k-egroup S such that $\mathcal{C}^+ \subseteq$

$S \subseteq (\mathcal{C}^+ \cup P)$ and there is no other k-egroup S' with $\mathcal{C}^+ \subseteq S' \subseteq (\mathcal{C}^+ \cup P)$ and $\text{eval}(S') > \text{eval}(S)$.

- Tie-breaking rule $\mathcal{F}$ belongs to the family $\mathcal{F}^{\text{eval}}_{\text{pess}}$ of *pessimistic tie-breaking rules* if and only if it always outputs some k-egroup S such that $\mathcal{C}^+ \subseteq S \subseteq (\mathcal{C}^+ \cup P)$ and there is no other k-egroup S' with $\mathcal{C}^+ \subseteq S' \subseteq (\mathcal{C}^+ \cup P)$ and $\text{eval}(S') < \text{eval}(S)$.

We remark that the definitions above come in two, substantially different variants. For each lexicographic tie-breaking rule, there is always exactly one egroup that will be returned by the rule for a particular set of pending candidates. However, this is not the case for the families of pessimistic and optimistic rules. In fact, there might be many possible egroups whose value, computed in terms of a respective evaluation variant, is exactly the same. Such a feature seems to contradict the idea of a tie-breaking rule that should not, by itself, introduce ties again. However, we claim that choosing an arbitrary equally-valued ("tied") egroup is a proper way to circumvent this problem. Indeed, according to a particular evaluation all egroups with the same value are indistinguishable from each other.

8.2.3 Limits of Lexicographic Tie-Breaking

From Definition 8.5, we can immediately derive that lexicographic tie-breaking is straightforward in the case of scoring-based multiwinner voting rules. For these rules, extending the set of the confirmed candidates by any of all possible subsets of the desired cardinality from the set of pending candidates yields a committee that is tied for winning in a given election. Thus, to apply lexicographic tie-breaking, it is enough to select the best pending candidates with respect to the given order. We remark that applying lexicographic tie-breaking may be more complicated for general multiwinner voting rules. The reason is that there might be mutual dependencies between candidates in the set of pending candidates. These can lead to a case when a committee composed of the confirmed candidates and some subset of the pending candidates is not winning under a given rule.

It remains to be clarified whether one can find a reasonable order of the pending candidates in order to model optimistic or pessimistic tie-breaking rules in a simple way. We show that this is possible for every $\mathcal{F}^{\text{eval}}_{\text{bhav}}$, $\text{eval} \in \{\text{util}, \text{candegal}\}$, $\text{bhav} \in \{\text{opt}, \text{pess}\}$, using the fact that in these cases we can safely assume that there is a single utility function (see Observation 8.1). On the contrary, there is a counterexample for $\text{eval} = \text{egal}$ and $\text{bhav} \in \{\text{opt}, \text{pess}\}$. On the way to prove these claims we need to formally define what it means that one family of tie-breaking

rules can be used to simulate another family of tie-breaking rules. To this end, we first define the equivalence between tie-breaking perspectives.

Definition 8.6. Let $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$ be a set of attribute tags (treated exactly as usual characters), $\mathcal{C}$ be a fixed set of candidates, $X = (\mathcal{C}^+, \mathcal{P}, k, U)$ and $\hat{X} = (\hat{\mathcal{C}}^+, \hat{\mathcal{P}}, \hat{k}, \hat{U})$ be tie-breaking perspectives over $\mathcal{C}$, where $\mathcal{C}^+ \subset \mathcal{C}$ and $\mathcal{P} \subseteq \mathcal{C}$. Then, X and $\hat{X}$ are:

- $\boxed{\mathbb{C}}$-*equivalent* if and only if $\mathcal{C}^+ = \hat{\mathcal{C}}^+$,
- $\boxed{\mathbb{P}}$-*equivalent* if and only if $\mathcal{P} = \hat{\mathcal{P}}$,
- $\boxed{\mathbb{K}}$-*equivalent* if and only if $k = \hat{k}$, and
- $\boxed{\mathbb{U}}$-*equivalent* if and only if $U = (\hat{U})$.

Additionally, for a subset $\mathcal{A}$ of symbols $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$, we say that X and X' are $\mathcal{A}$-equivalent if and only if, for each symbol $\mathbb{S} \in \mathcal{A}$, they are $\mathbb{S}$-equivalent.

One can easily verify that the equivalence notions from Definition 8.6, demonstrated in example Example 8.3, indeed meet the requirements of equivalence relations; thus, we can speak of equivalence classes of tie-breaking perspectives (with respect to a given equivalence notion).

> **Example 8.3.** Consider a set $\mathcal{C} = \{a, b, c, d, e, f\}$ of candidates, some fixed family U of utility functions evaluating the candidates, and two tie-breaking perspectives $X = \{\{a, b\}, \{c, d\}, 3, U\}$, $Y = \{\{a\}, \{b, c, d\}, 3, U\}$. Perspectives X and Y are $\boxed{\mathbb{K}}$-equivalent and $\boxed{\mathbb{U}}$-equivalent or, alternatively, $\{\boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-equivalent, but they are not equivalent with respect to other notions. Every tie breaking perspective $Z = \{\{a, b\}, \mathcal{P}, k, U'\}$—for arbitrary set $\mathcal{P} \subseteq \mathcal{C} \setminus \{a, b\}$, arbitrary size k of egroup, and arbitrary family U' of utility functions evaluating candidates in $\mathcal{C}$—is $\boxed{\mathbb{C}}$-equivalent to X.

Intuitively, the equivalence notions from Definition 8.6 categorize different tie-breaking perspectives according to their vital elements, like confirmed or pending candidates. This allows us to formally describe when (i.e., for each families of tie-breaking perspectives) different rules coincide in the outcomes they provide as shown in Example 8.4.

Example 8.4. Let $\mathcal{C} = \{a, b, c, d, e, f\}$ be a set of candidates. The following table defines a single utility function u, obtained from a family of utility functions for a set (the detailed description of manipulators is not important) as described in Observation 8.1.

$u(\cdot)$	a	b	c	d	e	f
	14	3	15	4	11	20

Let us consider two following sets of votes $\mathcal{V}_1$, and $\mathcal{V}_2$.

Votes in $\mathcal{V}_1$:

$a \succ e \succ c \succ d \succ b \succ f$

$a \succ c \succ b \succ d \succ e \succ f$

$a \succ b \succ d \succ b \succ c \succ f$

$b \succ c \succ a \succ f \succ b \succ d$

Votes in $\mathcal{V}_2$:

$a \succ e \succ c \succ d \succ b \succ f$

$a \succ c \succ b \succ d \succ e \succ f$

$f \succ b \succ d \succ e \succ c \succ a$

$f \succ d \succ e \succ a \succ b \succ c$

Assuming election $\mathcal{E}_1 = (\mathcal{C}, \mathcal{V}_1)$ and $\mathcal{E}_2 = (\mathcal{C}, \mathcal{V}_2)$, let us analyze the outcome of these two election applying $\mathcal{F}_{\text{lex}}$ and $\mathcal{F}^{\text{util}}_{\text{opt}}$ when seeking k-egroups for different values of k but always using k-Approval (i.e., t-Approval where the value of t always coincides with the size of the desired egroup). We define the order $\succ$ according to which rule $\mathcal{F}_{\text{lex}}$ breaks ties decreasingly with respect to the values of candidates according to u.

For election $\mathcal{E}_1$, we seek an egroup of size two. Hence, candidate a is the confirmed candidate and candidates b and c are the pending candidates. Thus, we have a tie-breaking perspective $X_1 = (\{a\}, \{b, c\}, 2, \{u\})$ for election $\mathcal{E}_1$. Applying $\mathcal{F}^{\text{util}}_{\text{opt}}$ to X_1, we obtain the winning k-egroup $\{a, c\}$ since it clearly is superior to $\{a, b\}$ with respect to the utilitarian value. Since $c \succ b$ according to $\succ$, rule $\mathcal{F}_{\text{lex}}$ also selects committee $\{a, c\}$.

Regarding election $\mathcal{E}_2$, we look for an egroup of size three. Thus, we obtain a tie-breaking perspective $X_2 = (\emptyset, \{a, b, c, d, e, f\}, 3, \{u\})$ (note, that X_2 is $\boxed{\mathbb{U}}$-equivalent to X_1). Now, tie-breaking rule $\mathcal{F}^{\text{util}}_{\text{opt}}$ clearly selects $\{a, c, f\}$. In fact, since candidates c, a, and f are in front of all other candidates in $\succ$, $\mathcal{F}_{\text{lex}}$ also selects $\{a, c, f\}$.

The coincidence of the outcomes of $\mathcal{F}_{\text{lex}}$ and $\mathcal{F}^{\text{util}}_{\text{opt}}$ in Example 8.4 is not accidental. In fact, because we chose $\succ$ that "matched" the utility function u and was independent of other features of tie-breaking perspectives (like pending

candidates, confirmed candidates, and the egroup size), the outcomes of both tie-breaking rules would be exactly the same for every other tie-breaking perspective that is $\boxed{\mathbb{U}}$-equivalent to perspectives X_1 and X_2. Thus, we could actually use them interchangeably for every tie-breaking perspective that consists of utility function u. In order to formally state this claim in Proposition 8.3, in the following definition we introduce the concept of simulating one family of rules by another family of rules.

Definition 8.7. For a nonempty set $\mathcal{A} \subseteq \{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$ of attribute tags, and for two tie-breaking families $\mathcal{F}$ and $\mathcal{F}'$, we say that $\mathcal{F}$ can *$\mathcal{A}$-simulate* $\mathcal{F}'$ if, for every nonempty set of candidates $\mathcal{C}$ and for every tie-breaking perspective X over $\mathcal{C}$, there exists a rule $F \in \mathcal{F}$ such that for each tie-breaking perspective in the equivalence class of X according to $\mathcal{A}$-equivalence there exists a rule $F' \in \mathcal{F}'$ such that F and F' yield the same output for this perspective. We call rule F an *$\mathcal{A}$-simulator*.

At first glance, Definition 8.7 might seem overcomplicated. However, it is tailored to grasp different facets of simulation. On the one hand, one can always find a lexicographic order and use it for breaking ties if all of the following are known: confirmed candidates, pending candidates, utility functions, and the size of an egroup; formally, the family of lexicographic tie-breaking rules $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-simulates every other family. Thus, one needs some flexibility in the definition of simulation to keep the definition expressive enough. On the other hand, it is somewhat clear that without fixing the utility functions, one cannot simulate optimistic or pessimistic tie-breaking rules.

Observation 8.2. *The family of lexicographic tie-breaking rules does not* $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}\}$*-simulate* $\mathcal{F}^{\text{eval}}_{\text{bhav}}$.

Proof. Suppose $k = 1$, $\mathcal{C}^+ = \emptyset$, and $\mathcal{P} = \{b_1, b_2\}$; that is, we are going to select either b_1 or b_2 who are tied. Let us fix a family $U \coloneqq \{u\}$ of utility functions such that $u(b_1) \coloneqq 1$ and $u(b_2) \coloneqq 0$. For the family U of utility functions clearly $\mathcal{F}^{\text{eval}}_{\text{opt}}$ selects candidate b_1. Now, consider a family $U' = \{u'\}$ of utility functions where u' assigns utility one to candidate b_2 and zero otherwise. For this family, $\mathcal{F}^{\text{eval}}_{\text{opt}}$ selects candidate b_2. This means that we cannot find a $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}\}$-simulator F from family $\mathcal{F}_{\text{lex}}$ of tie-breaking rules because in the first case F would have to choose b_1 and in the second case b_2 would have to be chosen. This is impossible using a single preference order over $\{b_1, b_2\}$. Similar families of functions (obtained by redefining u by exchanging each returned value one with value zero and vice versa) yield a proof for $\mathcal{F}^{\text{eval}}_{\text{pess}}$ as well. □

Next, we show that for some cases it is sufficient to fix just the utility functions in order to simulate optimistic or pessimistic tie-breaking rules (see Proposition 8.3). For other cases, however, one *has* to fix all of the following: confirmed candidates, pending candidates, utility functions, and the size of an egroup (see Proposition 8.4).

Proposition 8.3. *For every* $\text{eval} \in \{\text{util}, \text{candegal}\}$ *and* $\text{bhav} \in \{\text{opt}, \text{pess}\}$*, the family* $\mathcal{F}_{\text{lex}}$ *can* $\{\boxed{\mathbb{U}}\}$*-simulate* $\mathcal{F}^{\text{eval}}_{\text{bhav}}$*; additionally, for* m *candidates and* r *utility functions, a* $\{\boxed{\mathbb{U}}\}$*-simulator* $F \in \mathcal{F}_{\text{lex}}$ *can be found in* $O(m \cdot (r + \log m))$ *time.*

Proof. Recall from Observation 8.1 that if $\text{eval} \in \{\text{util}, \text{candegal}\}$, then there exists a single utility function u' that is equivalent to the given family of utility functions (with respect to the evaluation of egroup utilities). Hence, we compute such a function u' in $O(m \cdot r)$ time precisely following its definition as in the proof of Observation 8.1. We say an order $>_{\text{lex}}$ of the candidates is *consistent* with some utility function u if $c >_{\text{lex}} c'$ implies $u(c) \geq u(c')$ for optimistic tie-breaking and $c >_{\text{lex}} c'$ implies $u(c) \leq u(c')$ for pessimistic tie-breaking. Any lexicographic tie-breaking rule defined by an order $>_{\text{lex}}$ that is consistent with the utility function u' simulates $\mathcal{F}^{\text{eval}}_{\text{bhav}}$. We compute a consistent order by sorting the candidates according to u' in $O(m \cdot \log m)$ time. □

Proposition 8.3 describes an important feature of optimistic utilitarian and candidate-wise egalitarian tie-breaking and their pessimistic variants. Intuitively, the proposition says that for these tie-breaking mechanisms one can compute a respective linear order of candidates. Then one can forget all the details of the initial tie-breaking mechanism and use the order to determine winners. Importantly, the order can be computed a priori, without even knowing a multiwinner voting rules to be used and the votes in an election. Unfortunately, the simulation of pessimistic and optimistic egalitarian tie-breaking turns out to be more complicated. As we show in Proposition 8.4, simulating these tie-breaking rules by lexicographic tie-breaking always requires full information on a tie-breaking perspective the tie-breaking needs to be applied to.

Proposition 8.4. *For each nonempty set* $\mathcal{A} \subseteq \{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$ *of size at most three, the lexicographic tie-breaking family of rules does not* $\mathcal{A}$*-simulate* $\mathcal{F}^{\text{egal}}_{\text{bhav}}$ *assuming* $\text{bhav} \in \{\text{opt}, \text{pess}\}$*.*

Proof. From Observation 8.2 we already know that the family of lexicographic tie-breaking rules cannot $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}\}$-simulate the family of egalitarian pessimistic tie-breaking rules or the family of egalitarian optimistic tie-breaking rules.

Next, we build one counterexample for each of the remaining size-three subsets of $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$ to show our claim. To this end, let us fix a set of candidates $\mathcal{C} = \{b_1, b_2, m_1, m_2, o_1, o_2\}$ and a family $U = \{u_1, u_2\}$ of utility functions as depicted in the table below.

$u(\cdot)$	b_1	b_2	m_1	m_2	o_1	o_2
u_1	10	5	4	0	0	0
u_2	1	2	5	7	0	0

First, we prove that the family $\mathcal{F}_{\text{lex}}$ cannot $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulate $\mathcal{F}^{\text{egal}}_{\text{bhav}}$ for $\text{bhav} \in \{\text{opt}, \text{pess}\}$. Let us fix $\mathcal{C}^+ = \emptyset$, $\mathcal{P} = \mathcal{C} \setminus \{o_1, o_2\}$. We consider the optimistic variant of egalitarian tie-breaking for $k = 1$, so we are searching for a 1-egroup. Looking at the values of U, we see that candidate m_1 gives the best possible egalitarian evaluation value which is four. This means that a $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulator $F \in \mathcal{F}_{\text{lex}}$ has to use an order where m_1 precedes both b_1 and m_2. However, it turns out that if we set $k = 2$, then the best 2-egroup consists exactly of candidates b_1 and m_2. This leads to a contradiction because now candidates b_1 and m_2 should precede m_1 in F's lexicographic order. Consequently, family $\mathcal{F}_{\text{lex}}$ does not $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulate $\mathcal{F}^{\text{egal}}_{\text{opt}}$. Using the same values of utility functions and the same sequence of the values of k we get a proof for the pessimistic variant of egalitarian evaluation.

Second, we prove that the family $\mathcal{F}_{\text{lex}}$ cannot $\{\boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-simulate $\mathcal{F}^{\text{egal}}_{\text{bhav}}$ for $\text{bhav} \in \{\text{opt}, \text{pess}\}$. This time, we fix $\mathcal{P} = \mathcal{C} \setminus \{o_1, o_2\}$, $k = 2$. We construct the first case by setting $\mathcal{C}^+ = \{o_1\}$. Using the fact that in both functions candidate o_1 has utility zero, we choose exactly the same candidate as in the proof of $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulation for the case $k = 1$; that is, for the optimistic variant, the winning 2-egroup is m_1 and o_1. Consequently, m_1 precedes b_1 and m_2 in the potential $\{\boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-simulator's lexicographic order. Towards a contradiction, we set $\mathcal{C}^+ = \emptyset$. The situation is exactly the same as in the proof of the $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulation case. Now, the winning 2-egroup consists of b_1 and m_2 which completes the proof for the optimistic case. By almost the same argument, the result holds for the pessimistic variant.

Finally, we prove that the family $\mathcal{F}_{\text{lex}}$ cannot $\{\boxed{\mathbb{C}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-simulate $\mathcal{F}^{\text{egal}}_{\text{bhav}}$ for $\text{bhav} \in \{\text{opt}, \text{pess}\}$. We fix $\mathcal{C}^+ = \emptyset$, $k = 2$. For the first case we pick $\mathcal{P} = \{b_2, m_1, m_2\}$. The best egalitarian evaluation happens for the 2-egroup consisting of b_2 and m_1. This imposes that, in the potential $\{\boxed{\mathbb{C}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-simulator's order, b_2 and m_1 precede the remaining candidates (in particular, m_1 precedes m_2). However, for $\mathcal{P} = \mathcal{C}$ the best 2-egroup changes to that consisting of b_1 and m_2

which gives a contradiction (m_2 precedes m_1). As in the previous cases, the same argument provides a proof for the pessimistic variant. □

Proposition 8.4 implies that pessimistic and optimistic egalitarian tie-breaking cannot be, in general, simulated by lexicographic tie-breaking with an order precomputed in advance to an election. In terms of computational complexity, however, finding winners for pessimistic egalitarian tie-breaking remains tractable whereas the same task for optimistic egalitarian tie-breaking is intractable. We devote the next section to show this dichotomy as well as to establish computational hardness of computing winners for the other introduced tie-breaking rules.

8.3 Complexity of Tie-Breaking

It is natural to ask whether the tie-breaking rules proposed in Section 8.2.2 are practical in terms of their computational complexity. If not, then there is little hope for effective and efficient coalitional manipulation because tie-breaking might be an inevitable subtask to be solved by the manipulators. Indeed, manipulators might not be "powerful" enough to secure victory of their desired egroup completely avoiding tie-breaking.

Clearly, we can perform in linear time every lexicographic tie-breaking rule that is defined through some predefined order of the candidates. Hence, we focus on the rules that model optimistic or pessimistic manipulators. To this end, we analyze the following computational problem.

$\mathcal{F}^{\text{eval}}_{\text{bhav}}$-Tie-Breaking ($\mathcal{F}^{\text{eval}}_{\text{bhav}}$-TB)
eval $\in \{\text{util}, \text{egal}, \text{candegal}\}$, bhav $\in \{\text{opt}, \text{pess}\}$
Input: A set $\mathcal{C}$ of candidates partitioned into a set $\mathcal{P}$ of pending candidates and a set $\mathcal{C}^+$ of confirmed candidates, the size k of the excellence-group such that $|\mathcal{C}^+| < k < |\mathcal{C}|$, a family of manipulator utility functions $U = \{u_1, u_2, \ldots, u_r\}$ where $u_i \colon \mathcal{C} \to \mathbb{N}$, and an evaluation threshold $q \in \mathbb{N}$.
Question: Is there a size-k set $S \subseteq \mathcal{C}$ such that S is selected according to $\mathcal{F}^{\text{eval}}_{\text{bhav}}$, $\mathcal{C}^+ \subseteq S$, and $\text{eval}(S) \geq q$?

Naturally, we may assume that the number of candidates and the number of utility functions are polynomially upper-bounded in the size of the input. However, both the evaluation threshold and the utility function values are encoded in binary.

Note that an analogous problem has not been considered for single-winner elections. The reason behind this is that, for single-winner elections, optimistic and pessimistic tie-breaking rules can be easily simulated by lexicographic tie-breaking rules. To obtain the appropriate lexicographic tie-breaking rules, it is sufficient to order the candidates with respect to their value to manipulators. However, one cannot simply apply this approach for egroups, because there might be exponentially many different egroups to consider. Even if this exponential blow-up were acceptable, it would still be unclear how to derive an order of candidates from the computed values of egroups. Yet, using a different technique, we can simulate tie-breaking in multiwinner elections with a lexicographic tie-breaking rule for several variants of evaluation.

8.3.1 Utilitarian and Candidate-Wise Egalitarian Tie-Breaking

As a warm-up, we observe that tie-breaking can be applied and performed efficiently if the k-egroups are evaluated according to the utilitarian or candidate-wise egalitarian variant. The corresponding result follows almost directly from Proposition 8.3.

Corollary 8.5. *Let m denote the number of candidates and r denote the number of manipulators. Then, one can solve $\mathcal{F}^{\text{eval}}_{\text{bhav}}$-Tie-Breaking in $O(m \cdot (r + \log m))$ time for* $\text{eval} \in \{\text{util}, \text{candegal}\}$ *and* $\text{bhav} \in \{\text{opt}, \text{pess}\}$.

Proof. The algorithm works in two steps. First, it computes a lexicographic tie-breaking rule $\mathcal{F}_{\text{lex}}$ that simulates $\mathcal{F}^{\text{eval}}_{\text{bhav}}$ (recall Example 8.4) in $O(m \cdot (r + \log m))$ time as described in Proposition 8.3. Second, it applies tie-breaking rule $\mathcal{F}_{\text{lex}}$, and evaluates the resulting k-egroup in $O(k \cdot r)$ time. The running time of applying $\mathcal{F}_{\text{lex}}$ is linear with respect to the input length (see Section 8.2.3). □

8.3.2 Egalitarian Tie-Breaking

In this section, we consider the optimistic and pessimistic tie-breaking rules when applied to seeking a k-egroup evaluated according to the egalitarian variant. First, we show that applying and evaluating egalitarian tie-breaking is computationally easy for pessimistic manipulators but computationally intractable for optimistic manipulators even if the size of the egroup is small. Being pessimistic, the main idea is to "guess" the manipulator that is least satisfied and select the candidates appropriately. We show the computational worst-case hardness of the optimistic case via a reduction from Set Cover.

Theorem 8.6. *Let m denote the number of candidates, r denote the number of manipulators, q denote the evaluation threshold, and k denote the size of an egroup. Then, one can solve $\mathcal{F}^{\text{egal}}_{\text{pess}}$-Tie-Breaking in $O(r \cdot m \log m)$ time, but $\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking is NP-hard and W[2]-hard when parameterized by k even if $q = 1$ and every manipulator only gives either utility one or zero to each candidate.*

Proof. For the pessimistic case, it is sufficient to "guess" the least satisfied manipulator x by iterating through r possibilities. Then, select $k - |\mathcal{C}^+|$ pending candidates with the smallest total utility for this manipulator in $O(m \log m)$ time. Finally, comparing the k-egroup with the worst minimum satisfaction over all manipulators to the given lower bound q on the satisfaction level solves the problem.

We prove the hardness for the optimistic case giving a polynomial-time many-one reduction from the Set Cover problem which, given a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of subsets of a universe $X = \{x_1, x_2, \ldots, x_n\}$ and an integer h, asks whether there exists a family $\mathcal{S}' \subseteq \mathcal{S}$ of size at most h such that $\bigcup_{S \in \mathcal{S}'} S = X$. Set Cover is NP-hard and W[2]-hard with respect to parameter h [DF99]. Let us fix an instance $I = (X, \mathcal{S}, h)$ of Set Cover. To construct an $\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking instance, we introduce pending candidates $\mathcal{P} = \{c_1, c_2, \ldots, c_m\}$ representing subsets in $\mathcal{S}$ and manipulators $u_1, u_2, \ldots, u_n$ representing elements of the universe. Note that there are no confirmed and rejected candidates. Each manipulator u_i gives utility one to candidate c_j if set S_j contains element x_i and zero otherwise. We set the excellence-group size $k \coloneqq h$ and the threshold $q \coloneqq 1$.

Observe that if there is a size-k subset $S \subseteq \mathcal{P}$ such that $\min_{i \in [n]} \sum_{s \in S} u_i(s) \geq 1$, then there exists a family $\mathcal{S}'$—consisting of the sets represented by candidates in S—such that each element of the universe belongs to the set $\bigcup_{S \in \mathcal{S}'} S$. On the contrary, if we cannot pick a group of candidates of size k for which every manipulator's utility is at least one, then instance I is a "no" instance. This follows from the fact that for each size-k subset $S \subseteq \mathcal{P}$ there exists at least one manipulator u^* for whom $\sum_{s \in S} u^*(s) = 0$. This translates to the claim that there exists no size-h subset $\mathcal{S}' \subseteq \mathcal{S}$ such that all elements in X belong to the union of the sets in $\mathcal{S}'$.

The reduction is executable in polynomial time, so we obtain the NP-hardness; since we had $k = h$, $\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking is W[2]-hard when parameterized by the size k of an excellence-group. □

From the W[2]-hardness proof of Theorem 8.6, we learn that a small egroup size (alone) does not make $\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking computationally tractable even

for very simple utility functions. Next, using a parameterized reduction from the W[1]-complete Multicolored Clique problem [Fel+09], we show that there is still no hope for fixed-parameter tractability (under standard assumptions) even for the combined parameter "number of manipulators and egroup size"; intuitively, this parameter covers situations where few manipulators are going to influence an election for a small egroup.

Theorem 8.7. *Let k denote the size of an egroup and r denote the number of manipulators. Then, parameterized by $r + k$, $\mathcal{F}_{\text{opt}}^{\text{egal}}$-Tie-Breaking is* W[1]*-hard.*

Proof. We describe a parameterized reduction from the Multicolored Clique problem, which is W[1]-complete with respect to the number of colors [Fel+09]. In this problem, given an undirected graph $G = (V, E)$, a non-negative integer h, and a vertex coloring $\phi\colon V \to \{1, 2, \dots, h\}$, we ask whether graph G admits a colorful h-clique, that is, a size-h vertex subset $Q \subseteq V$ such that the vertices in Q are pairwise adjacent and have pairwise distinct colors. Without loss of generality, we assume that the number of vertices of each color is the same; this number is referred to as y in the following. Let (G, ϕ), where $G = (V, E)$, be a Multicolored Clique instance. Let $\mathcal{V}(i) = \{v_1^i, v_2^i, \dots, v_y^i\}$ denote the set of vertices of color $i \in [h]$, and let $E(i, j) = \{e_1^{i,j}, e_2^{i,j}, \dots, e_{|E(i,j)|}^{i,j}\}$, where $i, j \in [h]$, $i < j$, denote the set of edges that connect a vertex of color i to a vertex of color j. We disregard possible edges between vertices of the same color as they cannot be part of any clique.

Candidates We create one confirmed candidate c^* and $|V| + |E|$ pending candidates. More precisely: for each $\ell \in [y]$, we create one *vertex candidate* a_ℓ^i for each vertex $v_\ell^i \in v(i)$, $i \in [h]$ and, for each $i, j \in [h]$ such that $i < j$, we create one *edge candidate* $b_t^{i,j}$ for each edge $e_t^{i,j} \in E(i, j)$, $t \in [E(i, j)]$. We set the size k of the egroup to $h + \binom{h}{2} + 1$ and set the evaluation threshold $q \coloneqq y + 1$. Next, we describe the manipulators and explain the high-level idea of the construction.

Manipulators and the Main Idea Our construction will ensure that there is a k-egroup X with $c^* \in X$ and $\text{egal}(X) \geq q$ if and only if X contains h vertex candidates and $\binom{h}{2}$ edge candidates that encode a colorful h-clique. To this end, we introduce the following manipulators.

1. For each color $i \in [h]$, there is a *color manipulator* μ_i ensuring that the k-egroup contains a vertex candidate $a_{z_i}^i$ corresponding to a vertex of

color i. Herein, variable z_i denotes the id of the vertex candidate (resp. vertex) that is *selected* for color i.

2. For each $i, j \in [h]$ such that $i < j$, there is one *color pair manipulator* $\mu_{i,j}$ ensuring that the k-egroup contains an edge candidate $b^{i,j}_{z_{i,j}}$ corresponding to an edge connecting vertices of colors i and j. Herein, variable $z_{i,j}$ denotes the id of the edge candidate (resp. edge) that is *selected* for color pair $\{i, j\}$, $i < j$.

3. For each $i, j \in [h]$ such that $i \neq j$, there are two *verification manipulators* $\nu_{i,j}$, $\nu'_{i,j}$ ensuring that vertex $v^i_{z_i}$ is incident to edge $e^{i,j}_{z_{i,j}}$ if $i < j$ or incident to edge $e^{j,i}_{z_{j,i}}$ otherwise.

It is easy to verify that if there exists a k-egroup in agreement with the descriptions in the previous three points, then it consists of h selected vertex candidates of different colors and $\binom{h}{2}$ selected edge candidates that altogether represent a colorful h-clique.

Utility Functions Let us now describe how we can guarantee the intended roles of the manipulators introduced in Items 1 to 3 above using utility functions.

1. Color manipulator μ_i, $i \in [h]$, has utility y for the confirmed candidate c^*, utility one for each candidate corresponding to a vertex of color i, and utility zero for the remaining candidates.

2. Color pair manipulator $\mu_{i,j}$, $i, j \in [h]$, $i < j$, has utility y for the confirmed candidate c^*, utility one for each candidate corresponding to an edge connecting vertices of colors i and j, and utility zero for the remaining candidates.

3. Verification manipulator $\nu_{i,j}$, $i, j \in [h]$, $i \neq j$, has utility ℓ for candidate a^i_ℓ, $\ell \in [y]$, utility $q - \ell$ for each candidate corresponding to an edge that connects vertex v^i_ℓ to a vertex of color j, and utility zero for the remaining candidates.

4. Verification manipulator $\nu'_{i,j}$, $i, j \in [h]$, $i \neq j$, has utility $q - \ell$ for candidate a^i_ℓ, $\ell \in [y]$, utility ℓ for each candidate corresponding to an edge that connects vertex v^i_ℓ, to a vertex of color j, and utility zero for the remaining candidates.

Correctness We argue that graph G admits a colorful clique of size h if and only if there is a k-egroup X with $c^* \in X$ and $\text{egal}(X) \geq q$.

Suppose that there exists a colorful clique H of size h. Create the k-egroup X as follows. Start with $\{c^*\}$ and add every vertex candidate that corresponds to some vertex of H and every edge candidate that corresponds to some edge connecting vertices of H. Each color manipulator and color pair manipulator receives total utility $y+1$, because H contains, by definition, one vertex of each color and one edge connecting two vertices for each color pair. It is easy to verify that the verification manipulator $\nu_{i,j}$ must receive utility ℓ from a vertex candidate and utility $q-\ell$ from an edge candidate and that the verification manipulator $\nu'_{i,j}$ must receive utility $q-\ell$ from a vertex candidate and utility ℓ from an edge candidate. Thus, $\text{egal}(X) = q = y+1$.

Suppose that there exists a k-egroup $X \subseteq \mathcal{C}$ such that $\text{egal}(X) \geq q$. Since no color manipulator can achieve utility $y+1$ unless c^* belongs to the winning k-egroup, it follows that $c^* \in X$. Because each color manipulator μ_i receives total utility at least $y+1$, X must contain some vertex candidate $a^i_{z_i}$ corresponding to a vertex of color i for some $z_i \in [y]$; we say that X *selects* vertex $v^i_{z_i}$. Since each color pair manipulator $\mu_{i,j}$ receives total utility at least $y+1$, X must contain some edge candidate $b^{i,j}_{z_{i,j}}$ corresponding to an edge connecting a vertex of color i and a vertex of color j for some $z_{i,j}$; we say that X *selects* edge $e^{i,j}_{z_{i,j}}$. We implicitly assumed that each color manipulator and color pair manipulator contributes exactly one selected candidate to X. This assumption is true because there are exactly $k-1$ such manipulators and each needs to select at least one candidate; hence, X is exactly of the desired size. In order to show that the corresponding vertices and edges encode a colorful h-clique, it remains to show that no selected edge is incident to a vertex that is not selected. Assume towards a contradiction that X selects an edge $e^{i,j}_{z_{i,j}}$ and some vertex $v^i_{z_i} \notin e^{i,j}_{z_{i,j}}$. However, either verification manipulator $\nu_{i,j}$ or verification manipulator $\nu'_{i,j}$ receives the total utility at most $q-1$; a contradiction. □

Finally, devising a simple ILP formulation, one can show that $\mathcal{F}^{\text{egal}}_{\text{opt}}$-TB becomes fixed-parameter tractable when parameterized by the combined parameter "number of manipulators and number of different utility values" [BKN21]. Fixed-parameter tractability for this parameterization covers scenarios with few manipulators that have simple utility functions; in particular, when few voters have 0/1 utility functions. The following Theorem 8.8 shows that, while neither parameterization by the number of manipulators (Theorem 8.6) nor by the number of different utility values (Theorem 8.7) makes $\mathcal{F}^{\text{egal}}_{\text{opt}}$-TB fixed-parameter

tractable, this can be achieved by combining these two parameters.

Theorem 8.8 ([BKN21])**.** *Let u_{diff} denote the number of different utility values and r denote the number of manipulators. Then, parameterized by $r + u_{\text{diff}}$, $\mathcal{F}_{\text{opt}}^{\text{egal}}$-Tie-Breaking is fixed-parameter tractable.*

8.4 Complexity of Coalitional Manipulation

In the previous section, we have seen that breaking ties optimistically or pessimistically—an essential subtask to be solved by the manipulators in general—can become computationally challenging; in most cases, however, this problem turned out to be computationally easy. In this section, we move on to our full framework and analyze the computational difficulty of voting strategically for a coalition of manipulators. To this end, we formalize our central computational problem. Let $\mathcal{M}$ be a multiwinner voting rule and let $\mathcal{F}$ be a multiwinner tie-breaking rule.

> $\mathcal{M}$-$\mathcal{F}$-eval-Coalitional Manipulation ($\mathcal{M}$-$\mathcal{F}$-eval-CM)
> eval $\in \{\text{util}, \text{egal}, \text{candegal}\}$
> **Input:** An election $(\mathcal{C}, \mathcal{V})$, an egroup size $k < |\mathcal{C}|$, r manipulators represented by their utility functions $U = \{u_1, u_2, \ldots, u_r\}$ such that, for all $i \in [r]$, $u_i \colon \mathcal{C} \to \mathbb{N}$, and an evaluation threshold $q \in \mathbb{N}$.
> **Question:** Is there a size-r multiset W of manipulative votes over $\mathcal{C}$ such that a k-egroup $S \subset \mathcal{C}$ that wins the election $(\mathcal{C}, \mathcal{V} \cup W)$ under $\mathcal{M}$ and $\mathcal{F}$ results in $\text{eval}(S) \geq q$?

The $\mathcal{M}$-$\mathcal{F}$-eval-CM problem is defined very generally; namely, one can consider any multiwinner voting rule $\mathcal{M}$ (in particular, any single-winner voting rule is a multiwinner voting rule with $k = 1$). In this chapter, however, we focus on t-Approval; hence, from now on, we narrow down our analysis of $\mathcal{M}$-$\mathcal{F}$-eval-CM to the t-Approval-$\mathcal{F}$-eval-CM problem.

In line with our intention to model optimistic and pessimistic attitudes of manipulators, we require that the evaluation of an optimistic/pessimistic tie-breaking rule $\mathcal{F}$ is the same as that of the manipulator's. Indeed, only when this is the case the tie-breaking rule reflects that the manipulator's expect a certain (pessimistic or optimistic) outcome of an election in case of a tie. More formally, for every $\text{eval} \in \{\text{util}, \text{egal}, \text{candegal}\}$, we focus on variants of t-Approval$-\mathcal{F}-$eval$-$CM where $\mathcal{F} \in \{\mathcal{F}_{\text{lex}}, \mathcal{F}_{\text{opt}}^{\text{eval}}, \mathcal{F}_{\text{pess}}^{\text{eval}}\}$. The excluded problem variants, such as breaking ties according to the utilitarian variant, but

evaluating egroups by the manipulators in the egalitarian way, might indeed be occasionally relevant; for example, when ties are broken by a third party. However, it would be arguable to assume that the third party breaks ties in an election according to private utility functions of the manipulators. Hence, modeling such an external tie-breaking completely independent of the manipulators would require another utility function (in addition to the manipulators' utility functions) over the candidates used for tie-breaking; thus, we skip such scenarios as they are incompatible with our model. However, we always allow lexicographic tie-breaking which models scenarios where a tie-breaking rule is fixed, known to all voters and irrelevant of the manipulators' utility functions.

On the way to show our results, we use a restricted version of t-Approval-$\mathcal{F}$-eval-Coalitional Manipulation that we call t-Approval-$\mathcal{F}$-eval-Coalitional Manipulation *with consistent manipulators*. In this variant, the input stays the same, but all manipulators must cast exactly the same vote to achieve the objective.

To increase readability, we decided to represent manipulators by their utility functions. As a consequence, we frequently use, for example, u_1 referring to the manipulator itself, even if we do not care about the values of utility function u_1 at the moment of usage. In this section, we also stick to the term "voters" meaning the set $\mathcal{V}$ of voters of an input election. We never call manipulators "voters"; however, we speak about the manipulative votes they cast.

As for the encoding of the input of $\mathcal{M}$-$\mathcal{F}$-eval-CM, we use a standard assumption; namely, that the number of candidates, the number of voters, and the number of manipulators are polynomially upper-bounded in the size of the input. Analogously to $\mathcal{F}^{\text{eval}}_{\text{bhav}}$-Tie-Breaking, both the evaluation threshold and the utility function values are encoded in binary.

In the subsequent sections, we first focus on the (computationally simpler) utilitarian and candidate-wise egalitarian evaluation variants (Section 8.4.1) and then consider the egalitarian evaluation (Section 8.4.2).

8.4.1 Utilitarian & Candidate-Wise Egalitarian: Manipulation Is Tractable

We show that t-Approval-$\mathcal{F}$-eval-Coalitional Manipulation can be solved in polynomial time for any constant $t \in \mathbb{N}$, any eval $\in \{\text{util}, \text{candegal}\}$, and any $\mathcal{F} \in \{\mathcal{F}_{\text{lex}}, \mathcal{F}^{\text{eval}}_{\text{opt}}, \mathcal{F}^{\text{eval}}_{\text{pess}}\}$. Whereas in general, for an instance $\mathcal{I}$ with input size $\langle\mathcal{I}\rangle$, our algorithm requires $O(\langle\mathcal{I}\rangle^4)$ steps, for k-Approval (i.e., $t = k$), we give a better, quadratic-time algorithm (with respect to $\langle\mathcal{I}\rangle$).

In several proofs in Section 8.4.1 we use value of a candidate for manipulators

(coalition) and say that a candidate is more valuable or less valuable than another candidate. Although we cannot directly measure the value of a candidate for the whole manipulators' coalition in general, thanks to Observation 8.1 we can assume a single utility function when discussing the utilitarian and candidate-wise egalitarian variants. Thus, assigning a single value to each candidate is justified.

We start with an algorithm solving the general case of t-Approval-$\mathcal{F}$-eval-CM, $\text{eval} \in \{\text{util}, \text{candegal}\}$, $\mathcal{F} \in \{\mathcal{F}_{\text{lex}}, \mathcal{F}_{\text{opt}}^{\text{eval}}, \mathcal{F}_{\text{pess}}^{\text{eval}}\}$. The basic idea is to "guess" the lowest final score z of a member of the winning k-egroup and the least preferred (according to some a fixed, given lexicographic order over the candidates) candidate of the k-egroup that obtains final score z; there are at most polynomially many (with respect to the input size) pairs to be guessed. Then, the algorithm, in polynomial time (with respect to the input size), finds an optimal manipulation leading to a k-egroup represented by the guessed pair. At first glance it might seem that one can greedily find an optimal manipulation for a guessed pair. However, as we will show in the proof of Theorem 8.9, this task requires solving a small enough instance (i.e., with the total weight of items polynomially upper-bounded in the input size of the manipulation problem instance) a variant of the weakly NP-hard Knapsack problem [KPP04].

Theorem 8.9. *Let m denote the number of candidates, n the number of voters, k the size of a desired egroup, and r the number of manipulators. One can solve t-Approval-$\mathcal{F}$-eval-Coalitional Manipulation in $O(k^2m^2r(n+r))$ time for any* $\text{eval} \in \{\text{util}, \text{candegal}\}$ *and* $\mathcal{F} \in \{\mathcal{F}_{\text{lex}}, \mathcal{F}_{\text{opt}}^{\text{eval}}, \mathcal{F}_{\text{pess}}^{\text{eval}}\}$.

Proof. We prove the theorem for the lexicographic tie-breaking rule $\mathcal{F}_{\text{lex}}$. This is sufficient since, using Proposition 8.3, one can generalize the result to utilitarian and candidate-wise egalitarian variants. The basic idea of our algorithm is to fix certain parameters of a solution and then to reduce the resulting subproblem to a variant of the Knapsack problem with polynomial-sized weights. The algorithm iterates through all possible value combinations of the following two parameters:

1. The lowest final score $z < |\mathcal{V} \cup W|$ of any member of the k-egroup and
2. the candidate $\hat{c}$ with final score z such that c is the least preferred member of the k-egroup with respect to tie-breaking rule $\mathcal{F}_{\text{lex}}$.

For each combination of the parameters, the algorithm computes an optimal solution if it exists. In this case, an optimal solution is a manipulation leading

to an egroup that maximizes the utility for the manipulation among all egroups described by the parameters z and $\hat{c}$. The algorithm outputs "yes" if, among the solutions computed for all combinations of the parameters, there exists a manipulation resulting in an egroup that has at least the utility requested by the instance's input. Otherwise, the algorithm outputs "no."

To show how to compute an optimal manipulation for some combination of the parameters, let us fix some z and $\hat{c}$. We denote by $\mathcal{C}^+$ the set of candidates who get at least $z+1$ approvals from the non-manipulative votes or who are preferred to $\hat{c}$ according to $\mathcal{F}_{\text{lex}}$ and get exactly z approvals from the non-manipulative votes. Assuming that the combination of parameter values is correct, all candidates from $\mathcal{C}^+ \cup \{\hat{c}\}$ must belong to the k-egroup. Let $k^+ \coloneqq |\mathcal{C}^+|$. For sanity, we check whether $k^+ < k$, that is, whether candidate $\hat{c}$ can belong to the k-egroup if the candidate obtains final score z. We discard the corresponding combination of solution parameter values if the check fails. Next, we ensure that $\hat{c}$ obtains the final score exactly z. If $\hat{c}$ receives less than $z - r$ or more than z approvals from non-manipulative votes, then we discard this combination of solution parameter values. Otherwise, let $\hat{s} \coloneqq z - \text{score}_{\mathcal{V}}(\hat{c})$ denote the number of additional approvals candidate $\hat{c}$ needs in order to get final score z. Let $k^* \coloneqq k - k^+ - 1$ be the number of remaining (not yet fixed) members of the k-egroup. Let $s^* \coloneqq r \cdot t - \hat{s}$ be the number of approvals to be distributed to candidates in $\mathcal{C} \setminus \{\hat{c}\}$.

Now, the manipulators have to influence k^* further candidates to join the k-egroup (so far only consisting of $\mathcal{C}^+ \cup \{\hat{c}\}$) and distribute exactly s^* approvals in total to candidates in $\mathcal{C} \setminus \{\hat{c}\}$ but at most r approvals per candidate (as each manipulator contributes at most one approval per candidate). To this end, let $\mathcal{C}^*$ denote the set of candidates which can possibly join the k-egroup. For each candidate $c \in \mathcal{C} \setminus (\mathcal{C}^+ \cup \{\hat{c}\})$ it holds that $c \in \mathcal{C}^*$ if and only if

1. $z - r \leq \text{score}_{\mathcal{V}}(c) \leq z - 1$ if c is preferred to $\hat{c}$ with respect to $\mathcal{F}_{\text{lex}}$, or
2. $z - r + 1 \leq \text{score}_{\mathcal{V}}(c) \leq z$ if $\hat{c}$ is preferred to c with respect to $\mathcal{F}_{\text{lex}}$.

A straightforward idea is to select the k^* elements from $\mathcal{C}^*$ which have the highest values (that is, utility) for the coalition. However, there are two issues: First, s^* might be too small; that is, there are too few approvals to ensure that each of the k^* best-valued candidates gets the final score at least z (resp. at least $z+1$). Second, s^* might be too large; that is, there are too many approvals to be distributed so that there is no way to do this without causing unwanted candidates to get final score at least z (resp. at least $z+1$).

Fortunately, we can easily detect these cases and deal with them efficiently. In the former scenario we reduce the remaining problem to an instance of Exact k-item Knapsack—the problem in which, for a given set of items, their values and weights, and a knapsack capacity, we search for k items that maximize the overall value and do not exceed the knapsack capacity. In the latter case, we show that we can discard the corresponding combination of solution parameters.

First, if $s^* \leq r \cdot k^*$, then one can certainly distribute all s^* approvals (e.g., to the k^* candidates that will finally join the k-egroup). Of course, it could still be the case that there are too few approvals available to push the desired candidates into the k-egroup in a greedy manner. To solve this problem, we build an Exact k^*-item Knapsack instance where each candidate $c^* \in \mathcal{C}^*$ is mapped to an item. We set the weight of c^* to $z - \mathrm{score}_{\mathcal{V}}(c^*)$ if c^* is preferred to $\hat{c}$ with respect to $\mathcal{F}_{\mathrm{lex}}$ and otherwise to $(z+1) - \mathrm{score}_{\mathcal{V}}(c^*)$. We set the value of each $c^* \in \mathcal{C}^*$ to be equal to the utility that candidate c^* contributes to the manipulators. Now, an optimal solution (given the combinations of parameter values is correct) must select exactly k^* elements from $\mathcal{C}^*$ such that the total weight is at most s^*. This corresponds to Exact k-item Knapsack if we set our knapsack capacity to s^*. Furthermore, finding any such set with maximum total value leads to an optimal solution. Even if the final total weight s' of the chosen elements is smaller than s^*, we can transfer the Exact k-item Knapsack solution to the correct solution of our problem. The total weight corresponds to the number of approvals used. Thus, with the Exact k-item Knapsack solution we spend s' approvals. Then, we use the surplus $s^* - s'$ approvals to approve the chosen candidates even more; this is always possible because we assumed that $s^* \leq r \cdot k^*$. Clearly, approving the chosen candidates even more cannot prevent them from being selected to the winning egroup under t-Approval.

Second, if $s^* > r \cdot k^*$, then one can certainly ensure that each of the k^* most valued candidates from $\mathcal{C}^*$ achieves the final score at least z (resp. at least $z+1$). In many cases, it will not be a problem to distribute the remaining approvals; for example, one can safely spend up to r approvals for each candidate from $\mathcal{C} \setminus \mathcal{C}^*$, that is, to candidates that have no chance to get enough points to join the k-egroup or to candidates which are already fixed to be in the k-egroup. Furthermore, each candidate from $\mathcal{C}^*$ that is not among the k^* most valued candidates can be safely approved $z - \mathrm{score}_{\mathcal{V}}(c^*) - 1$ times (resp. $z - \mathrm{score}_{\mathcal{V}}(c^*)$ times) without reaching final score z (resp. $z+1$); we denote by s^+ the total number of approvals distributed in this way. So, if $s^* \leq s^+ + r \cdot k^*$ (note that we also assume $s^* > r \cdot k^*$), then we can greedily push the k^* most valued candidates from $\mathcal{C}^*$ into the k-egroup (spending $r \cdot k^*$ approvals) and then safely

distribute the remaining approvals within $\mathcal{C} \setminus \{\hat{c}\}$ as discussed. If $s^* > s^+ + r \cdot k^*$, then there is no possibility of distributing approvals in a way that $\hat{c}$ is part of the k-egroup. Towards a contradiction let us assume that $\hat{c}$ is part of the k-egroup obtained after distributing $s^+ + r \cdot k^* + 1$ approvals. This means that we spend all possible s^+ approvals so that $\hat{c}$ is not beaten and $r \cdot k^*$ approvals to push k^* candidates to the winning k-egroup. Giving one more approval to some candidate c' from $\mathcal{C}^*$ that is not yet in the k-egroup, by definition of $\mathcal{C}^*$ and s^+, means that the score of c' is enough to push $\hat{c}$ out of the final k-egroup; a contradiction. Consequently, for the case of $s^* > s^+ + r \cdot k^*$, we discard the corresponding combination of solution parameters.

As for the running time, the first step is sorting the candidates according to their values in $O(m(r + \log(m)))$ time. Then let us consider the running time of two cases $s^* \leq r \cdot k^*$ and $s^* > r \cdot k^*$ separately. In the former case, we solve Exact k-item Knapsack in $O(k^2mr)$ time by using dynamic programming based on analyzing all possible total weights of the selected items until the final value is reached [KPP04, Chapter 9.7.3][3] (note that the maximum possible total weight is upper-bounded by kr). If $s^* > r \cdot k^*$, then we approve at most m candidates which gives running time $O(m)$. Thus, we can conclude that the running time of the discussed cases is $O(k^2mr)$. Additionally, there are at most $n + r$ values of z and at most m choices of $\hat{c}$. Summarizing, we get the running time $O(k^2m^2r(n + r))$. □

Next, we show that k-Approval-$\mathcal{F}$-eval-CM (i.e., the special case of t-Approval-$\mathcal{F}$-eval-CM where $t = k$) can be solved in quadratic time, that is, much faster than the general variant of the problem. On our way to present this result, we first give an algorithm for t-Approval-$\mathcal{F}$-eval-CM with consistent manipulators. Then, we argue that it also solves k-Approval-$\mathcal{F}$-eval-CM. The algorithm "guesses" the minimum score among all members of the winning egroup and then (according to the tie-breaking method) selects the best candidates that can reach this score.

Proposition 8.10. *Let m denote the number of candidates, n denote the number of voters, and r denote the number of manipulators. Then one can solve* t-Approval-$\mathcal{F}$-eval-Coalitional Manipulation *with consistent manipulators in $O(m(m+r+n))$ time for any* $\text{eval} \in \{\text{util}, \text{candegal}\}$ *and* $\mathcal{F} \in \{\mathcal{F}_{\text{lex}}, \mathcal{F}_{\text{opt}}^{\text{eval}}, \mathcal{F}_{\text{pess}}^{\text{eval}}\}$.

[3] Kellerer, Pferschy, and Pisinger [KPP04] present dynamic programming based on all possible total *values* of items. However (what they also remark), these can be exchanged with all possible total *weights* of items leading to an algorithm with running time polynomial in the maximum weight of items.

Proof. Consider an instance of t-Approval-$\mathcal{F}_{\text{lex}}$-eval-CM with consistent manipulators with an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ where $\mathcal{C}$ is a candidate set and $\mathcal{V}$ is a collection of non-manipulative votes, r manipulators, an egroup size k, and a lexicographic order $>_{\text{lex}}$ used by $\mathcal{F}_{\text{lex}}$ to break ties. In essence, we introduce a constrained solution form called a *canonical solution* and argue that it is sufficient to analyze only this type of solutions. Then we provide an algorithm that efficiently seeks for an optimal canonical solution.

At the beginning, we observe that when manipulators vote consistently, then we can arrange the top t candidates of a manipulative vote in any order. Hence, the solution to our problem is a size-t subset (instead of an order) of candidates which we call a set of *supported candidates*; we call each member of this set a *supported candidate*. We now introduce a vital concept of the proof, the "strength" of the candidates.

Strength Order of the Candidates Additionally, we introduce a new order $>_{\text{s}}$ of the candidates. It sorts them descendingly with respect to the score they receive from voters and, as a second criterion, according to the position in $>_{\text{lex}}$. Intuitively, the easier it is for some candidate to be a part of a winning k-egroup, the higher is the candidate's position in $>_{\text{s}}$. As a consequence, we state Claim 8.11.

Claim 8.11. *Let us fix an instance of t-Approval-$\mathcal{F}_{\text{lex}}$-eval-CM with consistent manipulators and a solution X which leads to a winning k-egroup S. For every supported (resp. unsupported) candidate c, the following holds:*

1. *If c is part of the winning k-egroup, then every supported (resp. unsupported) predecessor of c, according to $>_s$, belongs to S and*
2. *if c is not part of the winning k-egroup, then every supported (resp. unsupported) successor of c, according to $>_s$, does not belong to S.*

Proof. Fix an instance of t-Approval-$\mathcal{F}_{\text{lex}}$-eval-CM with consistent manipulators and a solution X resulting in a winning k-egroup S. Let us consider the respective order $>_{\text{s}}$ over the candidates in the instance.

We first show that Statement 1 regarding supported candidates holds. According to the statement, fix some supported candidate $c \in S$ and let p be a predecessor of c (according to $>_{\text{s}}$). Towards a contradiction, assume that $p \notin S$. This implies that either (i) the score of p is smaller than the score of c or (ii) their scores are the same but $c >_{\text{lex}} p$. Let us focus on case (i). Both considered

candidates are supported by all manipulators (note that manipulators vote consistently). Thus, as a consequence of $p >_s c$, we have that the score of p is at least as high as the score of c; a contradiction. Next, consider case (ii), where p and c have the same scores. Consequently, the mutual order of c and p in $>_s$ is the same as their order in $>_{\text{lex}}$ (in other words, the order of c and p in $>_s$ does not depend on scores of c and p because those must be the same prior to any manipulation). Since $c >_{\text{lex}} p$, it follows that, by definition of $>_s$, it must hold that $c >_s p$; a contradiction again. Eventually, we obtain that p has to be part of S which completes the argument.

An analogous approach leads to proofs for the remaining three cases stated in the theorem. □ (Claim 8.11)

Claim 8.11 justifies thinking about $>_s$ as a "strength order"; hence, in the proof we use the terms *stronger* and *weaker* candidate. Using Claim 8.11, we can fix some candidate c as the weakest in the winning k-egroup and then infer candidates that have to be and that cannot be part of this k-egroup. To formalize this idea, we introduce the concept of a *canonical solution*.

Canonical Solutions Assuming the case where $k \leq t$, we call a solution X leading to a winning k-egroup S canonical if all candidates of the winning egroup are supported; that is, $S \subseteq X$. In the opposite case, $k > t$, solution X is canonical if $X \subset S$ and X is a set of the t weakest candidates in S. For the latter case, the formulation describes the solution which favors supporting weaker candidates first and ensures that no approval is given to a candidate outside the winning k-egroup.

Canonical solutions are achievable from every solution without changing the winning k-egroup. One cannot prevent a candidate from winning by supporting the candidate more because this only increases the candidate's score. Consequently, we can always transfer approvals to all candidates from the winning k-egroup. For the case $k > t$, we then have to rearrange the approvals in such a way that only the weakest members of the k-egroup are supported. However, such a rearrangement cannot change the outcome because, according to Claim 8.11, we can transfer an approval from some stronger candidate c to a weaker candidate c' keeping both of them in the winning k-egroup.

Dropped and Kept Candidates By the assumption that $k < m$, for every solution (including canonical solutions) we can always find the strongest candidate who is not part of the winning egroup. We call this candidate the

dropped candidate. Note that we use the strength order in the definition of the dropped candidate; this order does not take manipulative votes into account. Further applying the assumption that $t < m$, without loss of generality, we can assume that the dropped candidate is not a supported candidate. This holds true because if the dropped candidate is not in the winning k-egroup even if supported, then we can support any other candidate c (which must exist because $t < k$) without changing the winning k-egroup. Due to Claim 8.11, if c is not in the winning k-egroup, then, even after supporting, c (which is by definition weaker than the dropped candidate) cannot become a member of the k-egroup. Otherwise, supporting c clearly cannot prevent it from being a member of the winning k-egroup. Naturally, by definition of the dropped candidate, all candidates stronger than the dropped candidate are members of the winning k-egroup. We call these candidates *kept candidates*.

General Description of the Algorithm The algorithm for t-Approval-$\mathcal{F}_{\text{lex}}$-eval-CM with consistent manipulators iteratively looks for an optimal canonical solution for every possible (non-negative) number e of kept candidates (alternatively, the algorithm checks all feasible possibilities of choosing the dropped candidate). Then, the algorithm compares all solutions and picks one that is resulting in an egroup liked the most by the manipulators. Observe that $k - t \leq e \leq k$. The upper bound k is the consequence of the fact that each kept candidate is (by definition) in the winning k-egroup. Since all candidates except for kept candidates have to be supported to be part of the winning egroup, we need at least $k - t$ kept candidates in order to be able to complete the k-egroup.

Running Time To analyze the running time of the algorithm described in the previous paragraph, several steps need to be considered. At the beginning we have to compute values of candidates and then sort the candidates with respect to their value. This step runs in $O(rm + m \log m)$ time. Similarly, computing $>_s$ takes $O(tn + m \log m)$ time. Then, Procedure 8.1 (described in detail later in this proof) needs $O(m)$ steps to find an optimal canonical solution for some fixed number e of kept candidates. Finally, we have at most $t+1$ possible values of e. Adding the times up, together with the fact that $t < m$, we obtain the running time $O(m(m + r + n))$.

What Remains to Be Done Procedure 8.1 describes how to look for an optimal canonical solution for a fixed number t of kept candidates. First, partition the candidate set in the following way. By $\mathcal{C}^*$ we denote the kept

candidates (which are the top e candidates according to $>_s$). Consequently, the $(e+1)$-st strongest candidate is the dropped candidate; say c^*. For every value of e, the corresponding dropped candidate, by definition, is not allowed to be part of the winning egroup. Let

$$D \coloneqq \{c \in \mathcal{C} \setminus (\mathcal{C}^* \cup \{c^*\}) \mid (\mathrm{score}_\nu(c) + r > \mathrm{score}_\nu(c^*)) \vee (\mathrm{score}_\nu(c) + r = \mathrm{score}_\nu(c^*) \wedge c >_{\mathrm{lex}} c^*)\}$$

be the set of *distinguished candidates*. Each distinguished candidate, if supported, is preferred over c^* to be selected into the winning k-egroup. Consequently, the distinguished candidates are all candidates who can potentially be part of the winning k-egroup. We remark that to fulfill our assumption that the dropped candidate is not part of a winning egroup, it is obligatory to support at least $k-e$ distinguished candidates. Note that $\mathcal{C}^* \cup \{c^*\} \cup D \neq \mathcal{C}$ is possible. The remaining candidates cannot be part of the winning k-egroup under any circumstances assuming e kept candidates. Also, set D might consist of less than $k-e$ required candidates (which is the case when there are too few candidates that, after supported, would outperform c^*). If such a situation emerges, then we skip the respective value of t. Making use of the described division into c^*, D, and $\mathcal{C}^*$, Procedure 8.1 incrementally builds the set X of supported candidates associated with an optimal solution until all possible approvals are used. Observe that since $k < |\mathcal{C}|$ and $t < |\mathcal{C}|$, it is guaranteed that for $e = k$ Procedure 8.1 will return a feasible solution for e; in fact, this solution will always result in a winning egroup consisting of all e kept candidates (irrespective of D).

Detailed Description of the Algorithm Before studying Procedure 8.1 in detail, consider Figure 8.1 illustrating the procedure on example data. In line 1, the procedure builds set X of supported candidates using the $k-e$ best valued distinguished candidates. Since only the distinguished candidates might be a part of the winning k-egroup besides the kept candidates, there is no better outcome achievable. Then, in line 2, the remaining approvals, if they exist, are used to support kept candidates. This operation does not change the resulting k-egroup. Then Procedure 8.1 checks whether all t approvals were used; that is, whether $t = |X|$. If not, then there are exactly $t - |X|$ remaining approvals to use. Note that at this stage set X contains k supported candidates which correspond to the best possible k-egroup; however, without spending all approvals. Let us call this k-egroup S. It is possible that there is no way to spend the remaining $t - |X|$ approvals without changing the winning k-egroup S.

Procedure 8.1: A procedure for finding an optimal set of supported candidates.

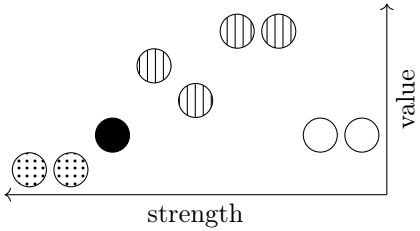
Input: Election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$; number t of approvals in t-Approval rule; size k of the winning k-egroup; a partition of $\mathcal{C}$ into kept candidates $\mathcal{C}^*$ (such that $e \coloneqq \left|\mathcal{C}^*\right|$ and $k - t \leq e \leq k$), a dropped candidate c^*, and distinguished candidates D (such that $|D| \geq k - e$).
Output: Optimal supported candidates set X

1 $X \longleftarrow \{k - e$ most valuable candidates from $D\}$
2 $X \longleftarrow X \cup \{\min\{e, t - |X|\}$ arbitrary candidates from $\mathcal{C}^*\}$
3 **if** $t \neq e|X|$ **then**
4 $\quad A \longleftarrow \{t - |X|$ weakest candidates from $\mathcal{C} \setminus X\}$
5 $\quad B \longleftarrow \{$top k strongest candidates from $X \cup A\}$
6 $\quad p \longleftarrow |B \setminus X|$
7 $\quad X \longleftarrow X \cup \{t - |X| - p$ weakest candidates from $\mathcal{C} \setminus X\}$
8 $\quad X \longleftarrow X \cup \{p$ most valuable candidates from $D \setminus X\}$
9 **return** X

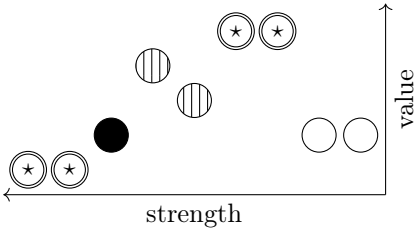
Then substitutions of candidates occur. The new candidates in the k-egroup can be only those that are distinguished and so far unsupported whereas the exchanged ones can be only so far supported distinguished candidates. This means that each substitution lowers the overall value of the winning k-egroup. So, the best what can be achieved is to find the minimal number of substitutions and then pick the most valuable remaining candidates from D to be substituted. The minimal number of substitutions can be found by analyzing how many candidates would be exchanged in the winning k-egroup if the weakest $t - |X|$ previously unsupported candidates were supported. The procedure makes such a simulation and computes the number p of necessary substitutions, in lines 4-6. Supporting the $t - |X| - p$ weakest unsupported candidates and then the p most valuable so far unsupported distinguished candidates gives the optimal k-egroup for e kept candidates (when all approvals are spent). Note that the number t of approvals is strictly lower than the number of candidates, so one always avoids supporting c^*.

Due to Proposition 8.3, the algorithm we presented can be applied also for pessimistic and optimistic evaluation because of the possibility of simulating these evaluations by a lexicographic order in time $O(m(r + \log(m)))$. □

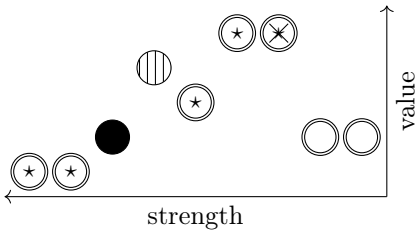
For k-Approval (i.e., where $t = k$), one can show that manipulators can always



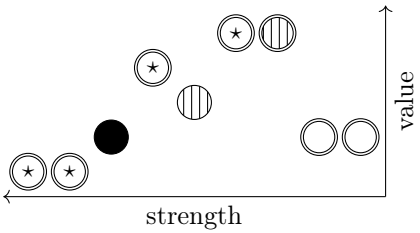
(a) The division of the candidates into the kept candidates (dotted), the dropped candidate (filled), the distinguished candidates (vertical lines), and the others who cannot be part of the winning egroup.



(b) An illustration of lines 1-2 of Procedure 8.1. The double-edged candidates form set X. The starred candidates would form the winning k-egroup if the double-edged candidates were supported.



(c) Supporting the weakest possible candidates to use all approvals. The winning egroup changes. The winners are marked with stars while the candidate which is not any more in the winning egroup is crossed out. Such a simulation is done in lines 4-6. As a result, the minimum number of substitutions in the k-egroup is computed.



(d) An illustration of the solution of the considered case computed by Procedure 8.1 in lines 7 to 8. One candidate from the k-egroup presented in Figure 8.1b has to be substituted; naturally, it is optimal to pick the most valuable possible candidate as a replacement for the substituted one. Supported candidates are double-edged and the winning k-egroup is starred.

Figure 8.1.: An illustrative example of a run of Procedure 8.1 for $e = 2$, nine candidates, 7-Approval, and 4-egroup. The horizontal position indicates the strength of a candidate—with the strength decreasing from left to right—and the vertical position indicates the value of a candidate. Since the number r of manipulators determines only the set of distinguished candidates, we do not specify r explicitly. We indicate the set of distinguished candidates instead. Figures 8.1a to 8.1d step by step present the execution of Procedure 8.1 on the way to find an optimal 4-egroup.

vote identically to achieve an optimal k-egroup [BKN21]. In a nutshell, for every egroup the manipulators can only increase the scores of the egroup's members by voting exactly for them. This fact leads to the following theorem.

Theorem 8.12 ([BKN21])**.** *Let m denote the number of candidates, n denote the number of voters, and r denote the number of manipulators. One can solve k-Approval-$\mathcal{F}$-*eval-Coalitional Manipulation *in $O(m(m+r+n))$ time for any* eval $\in \{\text{util}, \text{candegal}\}$ *and $\mathcal{F} \in \{\mathcal{F}_{\text{lex}}, \mathcal{F}_{\text{opt}}^{\text{eval}}, \mathcal{F}_{\text{pess}}^{\text{eval}}\}$.*

8.4.2 Egalitarian: Hard Even for Simple Tie-Breaking

In Section 8.3.2, we showed that already breaking ties might be computationally intractable. These intractability results only hold with respect to the egalitarian evaluation and optimistic manipulators. We now show that this intractability of $\mathcal{F}_{\text{opt}}^{\text{egal}}$-Tie-Breaking extends to coalitional manipulation for any tie-breaking rule and egalitarian evaluation. This includes the pessimistic egalitarian case which we consider to be highly relevant as it naturally models searching for a "safe" voting strategy.

Proposition 8.13. *For any tie-breaking rule $\mathcal{F}$, there is a polynomial-time many-one reduction from $\mathcal{F}_{\text{opt}}^{\text{egal}}$-Tie-Breaking to t-Approval-$\mathcal{F}$-*egal-Coalitional Manipulation.

Proof. We give a polynomial-time many-one reduction from $\mathcal{F}_{\text{opt}}^{\text{egal}}$-Tie-Breaking to t-Approval-$\mathcal{F}$-egal-Coalitional Manipulation; however, before we describe the actual reduction, we present a useful observation concerning $\mathcal{F}_{\text{opt}}^{\text{egal}}$-Tie-Breaking in the next paragraph.

Let us fix an instance $\mathcal{I}$ of $\mathcal{F}_{\text{opt}}^{\text{egal}}$-Tie-Breaking with a confirmed set $\mathcal{C}^+$, a pending set $\mathcal{P}$, a size k of an egroup, a threshold q, and a set of manipulators represented by a family U of utility functions. We construct a new equivalent instance $\mathcal{I}'$ of $\mathcal{F}_{\text{opt}}^{\text{egal}}$-Tie-Breaking with a larger set of manipulator utility functions $U' \supseteq U$. The construction is a polynomial-time many-one reduction which proves that we can "pump" the number of manipulators arbitrarily for instance $\mathcal{I}$. To add a manipulator, it is enough to set to q the utility that the manipulator gives to every candidate. Naturally, such a manipulator cannot have the total utility smaller than q, so the correct solution for $\mathcal{I}$ is also correct for $\mathcal{I}'$. Contrarily, when there is no solution for $\mathcal{I}$, it means that for every possible k-egroup S' there is some manipulator $\bar{u}$ such that $\text{egal}_{\bar{u}}(S') < q$. Consequently, one cannot find a solution for $\mathcal{I}'$ as well, because the set of possible k-egroups and their values of egalitarian utility do not change.

Now we can phrase our reduction from $\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking to t-Approval-$\mathcal{F}$-egal-Coalitional Manipulation. Let us fix an instance $\mathcal{I}$ of $\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking with a confirmed set $\mathcal{C}^+$, a pending set $\mathcal{P}$, a size k of an egroup, a threshold q, and a set U of r utility functions. Because of the observation about "pumping" instances of $\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking, we can assume, without loss of generality, that $t \cdot r \geq k - |\mathcal{C}^+|$ holds. In the constructed instance of t-Approval-$\mathcal{F}$-egal-CM equivalent to $\mathcal{I}$, we build an election that yields sets $\mathcal{P}$ and $\mathcal{C}^+$ and aim at an egroup of size k. However, it is likely that we need to add a set of dummy candidates that we denote by D. It is important to ensure that the dummy candidates cannot be the winners of the constructed election. To do so, we keep the score of each dummy candidate to be at most one, the score of each pending candidate to be $r+2$, and the score of each confirmed candidate to be at least $2r+3$. The construction starts from ensuring the scores of the confirmed candidates. Observe that in this step we add at most $(2r+3) \cdot |\mathcal{C}^+|$ voters (in case $t = 1$). If $t > |\mathcal{C}^+|$, then we have to add some dummy candidates in this step. We can upper-bound the number of the added dummy candidates by $((2r+3) \cdot |\mathcal{C}^+|)(t-1)$ (this bound is not tight). Analogously, we add new voters such that each pending candidate has score exactly $r+2$. At this step we have the election where we are able to spend $t \cdot r \geq k - |\mathcal{C}^+|$ approvals. We can select every possible subset of pending candidates to form the winning k-egroup by approving candidates in this subset exactly once. However, to be sure that we are able to distribute all approvals such that there is no tie, we ensure that the remaining $(t \cdot r) - (k - |\mathcal{C}^+|)$ approvals can be distributed to some candidates without changing the outcome. To achieve this goal, we add exactly $(t \cdot r) - (k - |\mathcal{C}^+|)$ dummy candidates with score zero. We set the evaluation threshold of the newly constructed instance to q.

By our construction, we are always able to approve enough pending candidates to form a k-egroup without considering ties, and we cannot make a dummy candidate a winner under any circumstances. Thus, if $\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking has a solution S, then we approve every candidate $c \in S$ such that c was in the pending set $\mathcal{P}$ before, and we obtain a solution to the reduced instance. In the opposite case, if there is no such a k-egroup whose egalitarian utility value is at least q, then the corresponding instance of t-Approval-$\mathcal{F}$-egal-Coalitional Manipulation also has no solution since the possible k-egroups are exactly the same. The reduction runs in polynomial time. □

Observe that the reduction proving Proposition 8.13 does not change the egroup size k. Additionally, the increase of the number of manipulators in the

resulting instances is polynomially upper-bounded in the egroup size k of input instances. This is due to the fact that even if we need to "pump" an initial instance to achieve $t \cdot r \geq k - |\mathcal{C}^+|$, then we add at most $\left\lceil \frac{k-|\mathcal{C}^+|}{t} \right\rceil \leq k$ manipulators. Thus, together with Theorem 8.6 and Theorem 8.7, Proposition 8.13 leads to the following theorem.

Theorem 8.14. *Let $\mathcal{F}$ be an arbitrary tie-breaking rule. Then, t-Approval-$\mathcal{F}$-egal-Coalitional Manipulation is* NP*-hard. Let r denote the number of manipulators, q denote the evaluation threshold and k denote the size of an egroup. Then, parameterized by $r + k$, t-Approval-$\mathcal{F}$-egal-CM is* W[1]*-hard. Parameterized by k, t-Approval-$\mathcal{F}$-egal-CM is* W[2]*-hard even if $q = 1$ and every manipulator only gives either utility one or zero to each candidate.*

Combining exhaustive enumeration of values describing essential properties of solutions and an extension of the ILP formulation from Theorem 8.8, we show that, for the combined parameter "number of manipulators and number of different utility values," fixed-parameter tractability of $\mathcal{F}_{\text{opt}}^{\text{egal}}$-Tie-Breaking extends to coalitional manipulation for both optimistic and pessimistic egalitarian tie-breaking.

Theorem 8.15. *Let r denote the number of manipulators and u_{diff} denote the number of different utility values. Parameterized by $r + u_{\text{diff}}$, t-Approval-$\mathcal{F}$-egal-Coalitional Manipulation with $\mathcal{F} \in \{\mathcal{F}_{\text{pess}}^{\text{egal}}, \mathcal{F}_{\text{opt}}^{\text{egal}}\}$ is fixed-parameter tractable.*

Proof. In a nutshell, we divide t-Approval-$\mathcal{F}_{\text{pess}}^{\text{egal}}$-egal-CM and t-Approval-$\mathcal{F}_{\text{opt}}^{\text{egal}}$-egal-CM into subproblems solvable efficiently in parameterized sense for the parameter under investigation. Then, we show that is it sufficient to consider only polynomially many subproblems to solve the problems.

The Main Idea We split the proof into two parts. In the first part, we define subproblems and show how to find a solution assuming that the subproblems are solvable in FPT time with respect to the parameter. In the second part, we show that, indeed, the subproblems are fixed-parameter tractable by providing their ILP formulations and applying Lenstra's result (Proposition 2.1). The inputs for t-Approval-$\mathcal{F}_{\text{pess}}^{\text{egal}}$-egal-CM and t-Approval-$\mathcal{F}_{\text{opt}}^{\text{egal}}$-egal-CM are the same, so let us consider an arbitrary input with an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ with n voters, m candidates, a size k of an excellence-group, and r manipulators represented

by a set $U = \{u_1, u_2, \ldots, u_r\}$ of their utility functions. Let u_{diff} be the number of different utility values.

An election resulting from a manipulation and a corresponding k-egroup emerging from the manipulation can be described by three non-negative integer parameters:

1. The lowest final score z of any member of the k-egroup;
2. the number p of *promoted candidates* from the k-egroup with a score higher than z which, at the same time, have score at most z without taking manipulative votes into consideration; and
3. the number b of *border candidates* with score z.

Observe that if as a result of a manipulation the lowest final score of members in a final k-egroup is z, then the promoted candidates are part of the k-egroup regardless of the tie-breaking method used. For border candidates, however, it might be necessary to run the tie-breaking rule to determine the k-egroup. In other words, border candidates become pending candidates unless all of them are part of the k-egroup. By definition, no candidate scoring lower than the border candidates is a member of the k-egroup; which gives border candidates their name. From now on, we refer to the election situation characterized by parameters z, p, b as a(n) *(input) state*. Additionally, we call a set of manipulator votes a *manipulation*.

Part 1: High-Level Description of the Algorithm For now, we assume that there is a procedure $\mathcal{Q}$ which runs in FPT time with respect to the combined parameter "number of manipulators and number of different utility values." Procedure $\mathcal{Q}$ takes values z, p, b and an instance of the problem as an input, and it finds a manipulation which leads to a k-egroup maximizing the egalitarian utility under either egalitarian optimistic or egalitarian pessimistic tie-breaking with respect to the input state. If such a manipulation does not exist, then procedure $\mathcal{Q}$ returns "no." The algorithm solving t-Approval-$\mathcal{F}^{\text{egal}}_{\text{pess}}$-egal-CM and t-Approval-$\mathcal{F}^{\text{egal}}_{\text{opt}}$-egal-CM invokes $\mathcal{Q}$ for all possible combinations of values z, p, and b. Eventually, it chooses the best manipulation returned by $\mathcal{Q}$ or returns "no" if $\mathcal{Q}$ always returned so. Since the value of z is at most $|\mathcal{V} + W|$ and b together with p are both upper-bounded by the number of candidates, we run $\mathcal{Q}$ at most $(n + r)m^2$ times. Because the input size grows polynomially with respect to the growth of the values r, m, and n, the overall algorithm runs in

FPT time with respect to the combined parameter "number of manipulators and number of different utility values."

Part 2: Basics and Preprocessing for the ILP To complete the proof, we describe procedure $\mathcal{Q}$ used by the above algorithm. In short, the procedure builds and solves an integer linear program that finds a manipulation leading to the state described by the input values. Before we describe the procedure in detail, we start with some notation. Fix some values of z, b, p and some election $E = (\mathcal{C}, \mathcal{V})$ that altogether form the input of $\mathcal{Q}$. For each candidate $c \in \mathcal{C}$, let a size-r vector $t = (u_1(c), u_2(c), \ldots, u_r(c))$, referred to as a *type vector*, define the *type* of c. We denote the set of all possible type vectors by $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$. Observe that $|\mathcal{T}| \leq u_{\mathrm{diff}}^r$. With each type vector t_i, $i \in [|\mathcal{T}|]$, we associate a set T_i consisting of all candidates of type t_i. We also distinguish the candidates with respect to their initial score compared to z. A candidate of type $t_i \in \mathcal{T}$, $i \in |\mathcal{T}|$, with score $z - j$, $j \in [r] \cup \{0\}$, belongs to group G_i^j. We denote all candidates with a score (excluding manipulative votes) higher than z by $\mathcal{C}^+$, whereas by $\mathcal{C}^-$ we denote the candidates with a score (excluding manipulative votes) strictly lower than $z - r$. For each type $t_i \in \mathcal{T}$ of a candidate, we define function $\mathrm{obl}(t_i) = |\mathcal{C}^+ \cap T_i|$, which gives the number of candidates of type t_i that are obligatory part of the winning k-egroup.

At the beginning, procedure $\mathcal{Q}$ tests whether the input values z, b, and p represent a correct state. From the fact that there has to be at least one candidate with score z, we get the upper bound $k - |\mathcal{C}^+| - 1$ for value p. To have enough candidates to complete the k-egroup, we need at least $k - |\mathcal{C}^+| - p$ candidates with score z after the manipulation which gives $b \geq k - |\mathcal{C}^+| - p$. Finally, the state is incorrect if the corresponding set $\mathcal{C}^+$ contains k or more candidates. If the input values are incorrect, then $\mathcal{Q}$ returns "no." Otherwise, $\mathcal{Q}$ continues with building a corresponding integer linear program. We give two separate integer linear programs—one for the optimistic egalitarian tie-breaking and the other one for the pessimistic egalitarian tie-breaking. Both programs consist of two parts. The first part models all possible manipulations leading to the state described by values z, p, and b. The second one is responsible for selecting the best k-egroup assuming the particular tie-breaking and considering all possible manipulations according to the first part. Although the whole programs are different from each other, the first parts stay the same. Thus, we postpone distinguishing between the programs until we describe the second parts. For the sake of readability, we present the ILP formulation step by step, describing each step in detail.

ILP: Common Part For each group G_i^j, $i \in [|\mathcal{T}|]$, $j \in [r] \cup \{0\}$, we introduce variables x_i^j and x_i^{j+} indicating the numbers of, respectively, border and promoted candidates from group G_i^j. Additionally, we introduce variables o and $\bar{o}$. The former represents the number of approvals used to get the obligatory numbers of border and promoted candidates. The latter indicates the number of approvals which are to be spent without changing the final k-egroup (thus, in some sense a complement of the obligatory approvals) resulting from the manipulation (e.g., approving candidates in $\mathcal{C}^+$, who are part of the winning k-egroup anyway, cannot change the outcome). We begin our integer linear program with ensuring that the values of x_i^j and x_i^{j+} are feasible:

$$x_i^{j+} + x_i^j \leq |G_i^j| \qquad \forall t_i \in \mathcal{T}, j \in [r] \cup \{0\}, \quad (8.1)$$

$$\sum_{t_i \in \mathcal{T}, j \in [r] \cup \{0\}} x_i^{j+} = p, \quad (8.2)$$

$$\sum_{t_i \in \mathcal{T}, j \in [r] \cup \{0\}} x_i^{j} = b, \quad (8.3)$$

$$x_i^{0+} + x_i^0 = |G_i^0| \qquad \forall t_i \in \mathcal{T}, \quad (8.4)$$

$$x_i^{r+} = 0 \qquad \forall t_i \in \mathcal{T}. \quad (8.5)$$

The constraints ensure that exactly p candidates are chosen to be promoted (8.2), exactly b candidates are selected to be border ones (8.3), and that, for every group, the sum of border and promoted candidates is not greater than the cardinality of the group (8.1). The last two constraint sets ensure that candidates who have score z are either promoted or border candidates (8.4) and that candidates with initial score $z - r$ cannot be promoted (i.e., get a score higher than z) (8.5). Next, we add the constraints concerning the number of approvals we need to use to perform the manipulation described by all variables x_i^j and x_i^{j+}. We start with ensuring that the manipulation does not exceed the number of possible approvals. As mentioned earlier, we store the number of required approvals using variable o.

$$o = \sum_{t_i \in \mathcal{T},\ j \in [r] \cup \{0\}} \left(x_i^j \cdot j + x_i^{j+} \cdot (j+1) \right), \quad (8.6)$$

$$o \leq tr. \quad (8.7)$$

Then, we model spending the $\bar{o}$ remaining votes (if any) to use all approvals.

$$\begin{aligned}\bar{o} \leq r(|\mathcal{C}^- \cup \mathcal{C}^+|) + \sum_{t_i \in \mathcal{T}} \sum_{j \in [r]} \left(|G_i^j| - x_i^j - x_i^{j+}\right)(j-1) \\ + \sum_{t_i \in \mathcal{T}, j \in [r]} \left(x_i^{j+} \cdot (r-j-1)\right),\end{aligned} \tag{8.8}$$

$$\bar{o} + o = tr. \tag{8.9}$$

The upper bound on the number of votes one can spend without changing the outcome presented in constraint (8.8) consists of three summands. The first one indicates the number of approvals which can be spent for candidates whose initial score was either too high or too low to make a difference in the outcome of the election resulting from the manipulation. The second summand counts the approvals we can spend for potential promoted and border candidates that eventually are not part of the winning k-egroup; we can give them less approvals than are needed to make them border candidates. The last summand represents the number of additional approvals that we can spend on the promoted candidates to reach the maximum of r approvals per candidate. This completes the first part of the ILP formulation in which we modeled the possible variants of promoted and border candidates for the fixed state (z, b, p).

ILP Extension for Optimistic Egalitarian Tie-Breaking In the second part, we find the final k-egroup by completing it with the border candidates according to the particular tie-breaking mechanism. Let us first focus on the case of the optimistic egalitarian tie-breaking. We introduce constraints allowing us to maximize the total egalitarian utility value of the final egroup; namely, for each group G_i^j, $i \in [|\mathcal{T}|]$, $j \in [r] \cup \{0\}$, we add a non-negative, integral variable $x_i^{j\bullet}$ indicating the number of border candidates of the given group chosen to be in the final k-egroup. The following constraints ensure that we select exactly $k - |\mathcal{C}^+| - p$ border candidates to complete the winning egroup and that, for each group G_i^j, we do not select more candidates than available.

$$\sum_{t_i \in \mathcal{T},\ j \in [r] \cup \{0\}} x_i^{j\bullet} = k - |\mathcal{C}^+| - p, \tag{8.10}$$

$$x_i^{j\bullet} \leq x_i^j \qquad \forall t_i \in \mathcal{T},\ j \in [r] \cup \{0\}. \tag{8.11}$$

To complete the description of the ILP formulation, we add the following final set of constraints defining the egalitarian utility s of the final k-excellence-group:

$$\sum_{t_i \in \mathcal{T},\ j \in [r] \cup \{0\}} t_i[q] \cdot (x_i^{j+} + x_i^{j\bullet}) + \sum_{t_i \in \mathcal{T}} t_i[q] \cdot \mathrm{obl}(t_i) \geq s \qquad \forall q \in [r]. \quad (8.12)$$

We set the goal of the program to maximize s and thus our program simulates the egalitarian optimistic tie-breaking.

ILP Extension for Pessimistic Egalitarian Tie-Breaking To solve a subproblem for the case of pessimistic egalitarian tie-breaking, we need a different approach. We start with an additional notation. For each type of candidate $t_i \in \mathcal{T}$, let $b_i = \sum_{j \in [r] \cup \{0\}} x_i^j$ denote the number of border candidates of this type. For each type $t_i \in \mathcal{T}$ and manipulator u_q, $q \in [r]$, we introduce a new integer variable d_i^q. Its value corresponds to the number of border candidates of type t_i who are part of the worst possible winning k-egroup according to manipulator u_q's preferences; we call these candidates the *designated candidates* of type t_i of manipulator u_q. For each variable d_i^q, we define a binary variable $\overline{\mathrm{used}}[d_i^q]$ which has value one if at least one candidate of type t_i is a designated candidate of manipulator u_q. Similarly, we define $\overline{\mathrm{fullyused}}[d_i^q]$ to indicate that all candidates of type t_i are designated by manipulator u_q. To give a program which solves the case of pessimistic egalitarian tie-breaking, we copy the first part of the previous integer linear program (constraints from (8.1) to (8.9)) and add new constraints. First of all, we ensure that each manipulator designates not more than the number of available border candidates from each type and that every manipulator designates exactly $k - p - |\mathcal{C}^+|$ candidates:

$$0 \leq d_i^q \leq b_i \qquad \forall t_i \in \mathcal{T}, q \in [r], \quad (8.13)$$

$$\sum_{t_i \in \mathcal{T}} d_i^q = k - p - |\mathcal{C}^+| \qquad \forall q \in [r]. \quad (8.14)$$

To force the semantics of the variables $\overline{\mathrm{used}}$—that is, a variable $\overline{\mathrm{used}}[d_i^q]$, $i \in [|\mathcal{T}|]$, $q \in [r]$, has value one if and only if variable d_i^q is at least one—we use the following constraints:

$$\overline{\mathrm{used}}[d_i^q] \leq d_i^q \qquad \forall t_i \in \mathcal{T}, q \in [r], \quad (8.15)$$

$$\overline{\mathrm{used}}[d_i^q] n \geq d_i^q \qquad \forall t_i \in \mathcal{T}, q \in [r]. \quad (8.16)$$

Similarly, for the variables $\overline{\text{fullyused}}$, we ensure that $\overline{\text{fullyused}}[d_i^q]$, $i \in [|\mathcal{T}|]$, $q \in [r]$, is one if and only if manipulator u_q designates all available candidates of type t_i with the subsequent constraints:

$$\overline{\text{fullyused}}[d_i^q] \geq 1 - (b_i - d_i^q) \qquad \forall t_i \in \mathcal{T}, q \in [r], \quad (8.17)$$

$$b_i - d_i^q \leq n(1 - \overline{\text{fullyused}}[d_i^q]) \qquad \forall t_i \in \mathcal{T}, q \in [r]. \quad (8.18)$$

Since our task is to perform pessimistic tie-breaking, we have to ensure that the designated candidates for each manipulator are the candidates whom the manipulator gives the least utility. We impose this by forcing that the more valuable candidates (for a particular manipulator) are used only when all candidates of all less valuable types (for the manipulator) are used (i.e., they are fully used). To achieve this we make use of the $\overline{\text{used}}$ and $\overline{\text{fullyused}}$ variables in the following constraint:

$$\overline{\text{used}}[d_i^q] \leq \overline{\text{fullyused}}[d_{i'}^q] \qquad \forall q \in [r] \cup \{0\},\ t_i, t_{i'} \in \mathcal{T} \colon t_i[q] > t_{i'}[q]. \quad (8.19)$$

Finally, we give the last set of constraints where s represents the pessimistic egalitarian k-egroup's utility, which our integer linear program wants to maximize:

$$\sum_{t_i \in \mathcal{T}} (d_i^q + \text{obl}(t_i)) \cdot t_i[q] \geq s \qquad \forall q \in [r]. \quad (8.20)$$

The ILP formulations, for both tie-breaking variants, use $O(ru_{\text{diff}}^r)$ variables. So, according to Proposition 2.1, we obtain fixed-parameter tractability with respect to the combined parameter $r + u_{\text{diff}}$. Consequently, procedure $\mathcal{Q}$ is in FPT with respect to the same parameter. □

Finally, applying the same technique as in Theorem 8.15 of solving a number of integer linear programs, one can obtain the following fixed-parameter tractability for t-Approval-$\mathcal{F}_{\text{lex}}$-egal-CM with respect to the number of manipulators plus the number of different utility values [BKN21].

Theorem 8.16 (Bredereck, Kaczmarczyk, and Niedermeier [BKN21])**.** *Let r denote the number of manipulators and u_{diff} denote the number of different utility values. Then, t-Approval-$\mathcal{F}_{\text{lex}}$-egal-*Coalitional Manipulation *parameterized by $r + u_{\text{diff}}$ is fixed-parameter tractable.*

8.5 Experimental Insights

Studying the computational complexity of $\mathcal{F}$-TB and t-Approval-$\mathcal{F}$-eval-CM, we devised several algorithms showing either polynomial-time computability or fixed-parameter tractability. Our goal for this section is to verify whether our algorithmic results dealing with t-Approval-$\mathcal{F}$-eval-CM are applicable in practice for small-sized and mid-sized elections (up to 40 candidates and up to 100 voters) in which coalitional manipulation is probably the most likely to happen in practice. Indeed, for a large election it might be hard to form and coordinate a group of manipulators large enough to have an influence on the outcome. Due to our focus on the running time and due to the preliminary nature of our experiments, we decided to synthetically generate elections only according to the Impartial Culture model.

For the sake of brevity, we do not present any running-time experiments for the problem of tie-breaking. Observe that, in general, tie-breaking is an unavoidable subtask of t-Approval-$\mathcal{F}$-eval-CM. Thus, the positive results that we obtained for coalitional manipulation (that are described further in this section), already provide positive results for the algorithms for tie-breaking.

We especially focus on the algorithms presented in Theorems 8.9 and 8.15 for the following two reasons. First, both algorithms significantly differ in terms of techniques they use—one of them is a purely combinatorial algorithm while the other one heavily relies on solving integer linear programs. We study how this difference, in practice, influences the running times of the algorithms. Second, the theoretical running times of both algorithms suggest that their practicality might be easily challenged by large instances. Consider an instance $\mathcal{I}$ of t-Approval-$\mathcal{F}$-eval-CM with r manipulators, n voters, m candidates and the number u_{diff} of different utility values that the manipulators assign to the candidates; let $\langle\mathcal{I}\rangle \in O((n + r\log(u_{\text{diff}}))m)$ be the (input) size of $\mathcal{I}$. Recall that the (worst-case) running time of the algorithm from Theorem 8.9 is $O(\langle\mathcal{I}\rangle^4)$, and the (worst-case) running time of the algorithm from Theorem 8.15 is at least $O(u_{\text{diff}}^r)^{2u_{\text{diff}}^r}\langle\mathcal{I}\rangle)$ (due to Proposition 2.1). Thus, even for very small instances, say four manipulators that report four different values of the utility values, the algorithm from Theorem 8.15 needs at least the enormous number of roughly 2^{128} steps. Obviously, in theory, the algorithm from Theorem 8.9 is undoubtedly more efficient with respect to the worst-case running time and appears to have practically infeasible running times only when the input consists rather of hundreds of candidates, manipulators and voters.

Conducting our experiments, we aim at verifying the practical applicability

of the analyzed algorithms and compare the running times they achieve. In our experiments we primarily focused on the running times of the studied algorithms with respect to the different quantities that, according to our theoretical analysis, have an influence on the computational efficiency. We note that, in particular, we did not conduct qualitative experiments on manipulation and that our experiments should rather be treated as preliminary and exploratory tests (we refer to the bachelor's thesis of Kalkbrenner [Kal19] for more involved experiments, including real-world data).

8.5.1 Experimental Setup

We conducted the experiments for two variants of the coalitional manipulation problem. Specifically, we tested the algorithm from Theorem 8.9 for t-Approval-$\mathcal{F}_{\text{opt}}^{\text{util}}$-util-CM and the algorithm from Theorem 8.15 for t-Approval-$\mathcal{F}_{\text{opt}}^{\text{egal}}$-egal-CM; both for two different values of t.

We generated two series of experiments. In the first series, which we call the *small series*, we fixed the egroup size $k = 5$ and the 3-Approval election rule, whereas in the second series, the *big series*, we chose $k = 10$ and 6-Approval. We partitioned each series' instances, all with elections consisting of 100 non-manipulative voters, into two collections. In one collection we varied the number of candidates from $k+1$ (k is the proper parameter value of the series) to 40 and in the other one we swept through different values of the number of manipulators between 10 and 40. When we were not changing the number of candidates we fixed it to 20 and when we were not changing the number of manipulators we fixed it to 10. We chose the rule and the size of the desired k-egroup in a way to avoid the k-Approval rule (where the number of approvals coincides with the k-egroup size) as it can be solved by the algorithm from Theorem 8.12. For both tested algorithms, we repeated all experiments 30 times and averaged the collected results.

To obtain a single instance of t-Approval-$\mathcal{F}$-eval-CM, we first generated an election according to the Impartial Culture model (Definition 6.7). To obtain the desired number of manipulators, for each election, we then uniformly at random drew the manipulators from the election voters. Then, for each manipulator, we constructed the utility function that assigns each candidate its Borda score in the ranking of the manipulator. Finally, we removed the votes of the manipulators from the original election obtaining a new election that, together with the utility functions constructed for the manipulators, formed a single instance.

Considering our choice of parameters, we decided to choose 100 voters, which is a common assumption in the multiwinner election literature [Bre+19a, Fal+18a,

Szu+20]. Considering up to 25 manipulators, we covered scenarios where the ratio between the number of non-manipulative voters and the manipulators varies from one to 25%. Cases in which there are more than 25% manipulators (compared with the number of non-manipulators) seem to be unlikely in practice due to a large number of manipulators. We chose the egroup size to be a half and a quarter of the candidates (in the basic scenario, where the number of candidates did not vary) to show different levels of "competitiveness." We avoided considering too many candidates and too high numbers of approvals because in practice, for humans, selecting (and ordering) too many candidates quickly becomes tedious as the number of possible options grows. At the same time, we did not want to consider very small elections; thus, we chose 20 candidates and rather a small numbers of approvals: three and six. However, we stress that properly selecting the election parameter values, especially the number of candidates, the size of a desired egroup, and the number of approvals t in t-Approval, should be further investigated in more detail to draw meaningful qualitative conclusions about manipulation. This applies also for the utility functions of manipulators, which we generated using Borda scores. We decided on Borda scores because of their simplicity and intuitiveness, but we believe that other ways of generating utility functions should be considered. For example, we expect that the running times could drop if one constrains the utility functions to take only natural values between one and three.

For the specification of the machines on which we conducted our experiments see Section 2.7. We used the Python Programming Language (specifically, the standard interpreter in version 3.6.9). To solve integer linear programs we used Gurobi Optimizer (version 8.1.1 build v8.1.1rc0). The code is publicly accessible online on GitHub (https://github.com/kalkbrennerei/maniplib).

8.5.2 Results and Interpretation

The obtained running times—illustrated in Figure 8.2 for the algorithm from Theorem 8.9 and in Figure 8.3 for the algorithm from Theorem 8.15—revealed that our algorithms achieved practically feasible running times. For the highest number 40 of candidates, we were able to solve all instances within around 150 seconds. For 25 manipulators, however, the worst running time we obtained was around 225 seconds. Thus, the results strongly suggest that we provided practically applicable algorithms that can solve coalitional manipulation problems for mid-sized elections.

Even though the theoretical upper bound of the running time of the algorithm from Theorem 8.15 is much worse, the corresponding approach was significantly

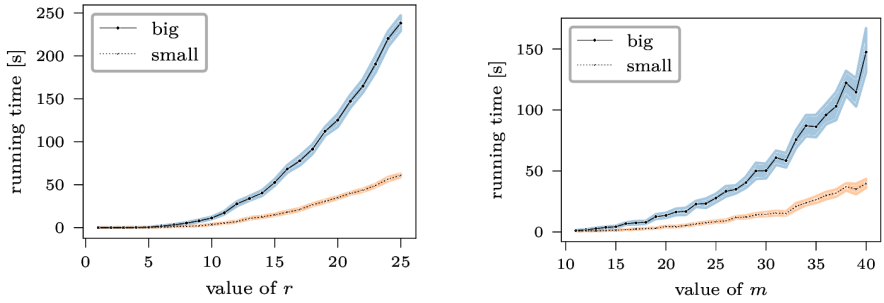


Figure 8.2.: Running times of the algorithm from Theorem 8.9 for different numbers r of manipulators and m of candidates. The marked lines depict the average running times over 30 trials for the big series (the dot-marked solid line) and the small series (the cross-marked dashed line). The lighter "bands" around the lines indicate the maximum and the minimum running times observed.

faster (roughly two times for the series with varying number of manipulators and up to 50 times for the series with varying number of candidates). This observation does not come as a surprise as integer linear solvers are nowadays highly optimized.

The experiments also clearly confirm the theoretical dependence on the number of candidates of the running times of the studied algorithms; namely the quadratic dependence for the algorithm from Theorem 8.9 and the linear dependence of the algorithm from Theorem 8.15. Our results regarding the algorithm from Theorem 8.15 also confirm our computational complexity analysis of this algorithm with respect to the number of manipulators. Indeed, on the right plot in Figure 8.3, we observe an exponential increase of the running time for increasing numbers of manipulators. The observed behavior of the running time with respect to the number of manipulators is also complaint with the theoretical running time given in Theorem 8.9. Indeed, both the experiments and the theoretical analysis suggest a quadratic dependence.

Last but not least, our preliminary experimental studies open up an avenue for further research on manipulation in multiwinner elections. There is a broad range of challenging open questions of at least two types.

First, what is the influence of the election structure on the running time? Observe that, somehow counterintuitively, the right plot in Figure 8.3 shows

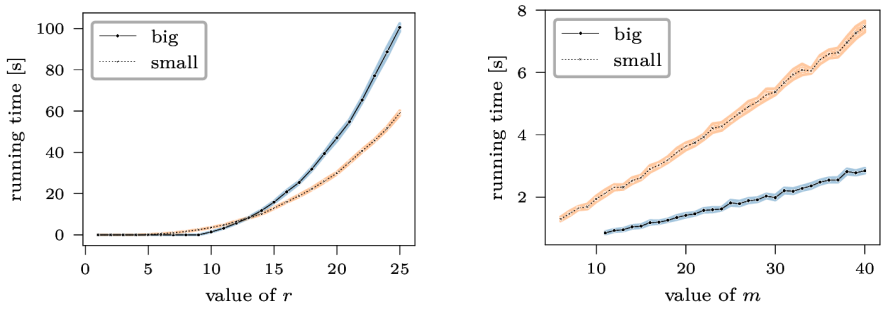


Figure 8.3.: Running times of the algorithm from Theorem 8.15 and m of candidates. The marked lines depict the average running times over 30 trials for the big series (the dot-marked solid line) and the small series (the cross-marked dashed line). The lighter "bands" around the lines indicate the maximum and the minimum running times observed.

that we obtained significantly worse running times when increasing the number of candidates for the small series than we did for the big series. The differences between these series were only in the size of the desired k-egroup and the number of approvals, so they had the same number of manipulators. We conjecture that the reason for these results is that it is "harder" to "satisfy" each manipulator (note that in Figure 8.3 we considered the egalitarian variant) with a smaller k-egroup. However, a more detailed study would be needed to confirm our conjecture. A further, somewhat related question, concerns the influence of the ratio between manipulators and the voters on the running time. Intuitively, the more manipulators, the simpler it is to successfully manipulate but, on the contrary, the harder it is to find a set of candidates satisfying the manipulators.

The second type of challenging open questions concerns the possible influence the manipulators can achieve, or wish to achieve, depending, for example, on their number. On the one hand, for many diverse manipulators, the most satisfying outcome can be very "near" to the current one because many outcomes can be similarly satisfying to a heterogeneous group of manipulators. On the other hand, a small group of similarly-minded manipulators can have a precise goal, yet be short of "power" to affect the outcome in a favorable way.

In summary, by devising practically efficient algorithms, we believe to have provided algorithmic tools that allow for exploring these two mentioned paths of

open questions. In particular, further experiments should consider other models of generating elections than Impartial Culture to simulate different levels of similarity of the manipulators' preferences. Looking from this perspective, the Mallows model (Definition 6.8) is, for example, suitable to model similarly-minded manipulators.

8.6 Conclusion

We developed a new model for and started a first systematic study of coalitional manipulation for multiwinner elections. Our analysis revealed that multiwinner coalitional manipulation requires models which are significantly more complex than those for single-winner coalitional manipulation or multiwinner non-coalitional manipulation. As described in the introduction, our model assumes that a given coalition of manipulators can compensate their (potential) utility loss after a manipulation in some way. Thus, in particular, our model does not analyze the dynamics of a coalition but rather it tries to assess its potential and possible influence.

In our work, on the one hand, we generalized tractability results for coalitional manipulation of t-Approval [CSL07, Lin11] and for non-coalitional manipulation of k-Approval [Mei+08, OZE13] to tractability of coalitional manipulation of t-Approval in case of utilitarian or candidate-wise egalitarian evaluation of egroups. On the other hand, we showed that coalitional manipulation becomes intractable in case of egalitarian evaluation of egroups.

Let us discuss a few findings in more detail (Table 8.1 surveys all our results). We studied lexicographic, optimistic, and pessimistic tie-breaking and showed that, with the exception of egalitarian group evaluation, winning excellence-groups can be determined very efficiently. The intractability (NP-hardness, parameterized hardness in form of W[1]- and W[2]-hardness) for the egalitarian case, however, turns out to hold even for quite restricted scenarios. We also demonstrated that numerous tie-breaking rules can be "simulated" by (carefully chosen) lexicographic tie-breaking, again except for the egalitarian case. Observe that the hardness of egalitarian tie-breaking holds only for the optimistic case while for the pessimistic case it is efficiently solvable. Hardness for the egalitarian optimistic scenario, however, translates into hardness results for coalitional manipulation *regardless* of the specific tie-breaking rule. On the contrary, coalitional manipulation becomes tractable for the other two evaluation strategies—"candidate-wise" egalitarian and utilitarian. Additionally, for few manipulators and few different utility values the manipulators assign to the

$\mathcal{F}^{\text{eval}}_{\text{bhav}}$-Tie-Breaking, easy cases:

settings (evaluation, behavior)	complexity	reference
utilitarian or cand.wise egalitarian, optimistic or pessimistic	$O(m \cdot (r + \log m))$	Cor. 8.5 †
egalitarian, pessimistic	$O(r \cdot m \log m)$	Th. 8.6

$\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking (egalitarian, optimistic):

parameters, restrictions	complexity	reference
general	NP-hard	Th. 8.6
k, 0/1 utilities and $q = 1$	W[2]-hard	Th. 8.6
$r + k$	W[1]-hard	Th. 8.7
$r + u_{\text{diff}}$	FPT	Th. 8.8

t-Approval-$\mathcal{F}$-eval-Coalitional Manipulation
utilitarian/cand.wise egalitarian, optimistic/pessimistic:

restrictions	complexity	reference
general	$O(k^2m^2r(n+r))$	Th. 8.9 ⋄
consistent manipulators	$O(m(m+r+n))$	Pr. 8.10 ⋄
$t = k$	$O(m(m+r+n))$	Th. 8.12 ⋄

t-Approval-$\mathcal{F}$-eval-Coalitional Manipulation
egalitarian, optimistic/pessimistic:

parameters, restrictions	complexity	reference
general	NP-hard	Th. 8.14 ⋄
k, 0/1 utilities and $q = 1$	W[2]-hard	Th. 8.14 ⋄
$r + k$	W[1]-hard	Th. 8.14 ⋄
$r + u_{\text{diff}}$	FPT	Th. 8.15 and Th. 8.16 ⋄

Table 8.1.: Computational complexity of tie-breaking and coalitional manipulation. Our results for t-Approval hold for any $t \geq 1$, and thus cover SNTV. The parameters are the size k of the egroup, the number r of manipulators, and the number u_{diff} of different utility values. Furthermore, m is the number of candidates and n is the number of voters. The result marked with † holds for all possible combinations of the respective evaluation and behavior variants. The results marked with ⋄ hold also for $\mathcal{F} = \mathcal{F}_{\text{lex}}$.

candidates, manipulation becomes tractable also for the egalitarian optimistic scenario.

Our study provides a handful of efficient algorithms that, as we have shown in Section 8.5, allow for further experimental study of coalitional manipulability. Among many issues that such a study can address, there are a few particularly remarkable ones like "Is finding a successful manipulation hard in practice?", "How likely is a successful manipulation?", and "How much, in practice, can an election outcome be affected by a coalition?" (see a book chapter by Conitzer and Walsh [CW16] addressing these questions in the single-winner case in both theory and practice). It appears to be interesting to apply the algorithms we provided to experimentally answer the mentioned questions. Furthermore, we believe that our algorithms may also serve for assessing different measures of robustness of an election as discussed in Section 7.6.

In our study, we entirely focused on shortlisting as one of the simplest tasks for multiwinner elections to analyze our evaluation functions. It is interesting and demanding to develop models for multiwinner rules that aim for proportional representation or diversity. For shortlisting, extending our studies to non-approval-like scoring-based voting correspondences would be a natural next step. In this context, already seeing what happens if one extends the set of individual scores from being only zero or one to more (but few) numbers is of interest. Moreover, we focused on deterministic tie-breaking mechanisms, ignoring randomized tie-breaking—another issue for future research.

An analysis of the dynamics of manipulator coalitions directing towards game theory seems promising as well. This is even more so since we identified practically applicable algorithms for several variants of coalitional manipulation. One interesting question is what would be a "best" coalition to create or join for a manipulator when assuming that the manipulator knows the preferences of all other voters. Another one is, for example, whether a particular coalition is stable. Intuitively, the utility for every voter that is a part of the manipulating coalition should not be below the utility the voter receives when voting sincerely. This is of course only a necessary condition to ensure the stability of a coalition. A deeper analysis of stability needs to consider game-theoretic aspects such as Nash or core stability [Nis+07].

CHAPTER 9

Conclusion

For nearly all of the issues we studied, we successfully achieved all three goals posed in the introduction (Chapter 1): bringing already studied concepts closer to reality, providing algorithmic tools to tackle the then-emerging computational problems, and conducting experimental studies based on the outcome of the preceding two goals. More specifically, first, we stepped forward towards capturing real-life scenarios better by coming up with new models or by extending existing ones. Second, for each of the models we have depicted the landscape of classical and parameterized computational complexity. According to the second goal, we also identified several special cases allowing for efficient solutions. Third, for all but one model, we performed a short, preliminary experimental study. Doing so, we mainly provided more insights into practical running times of our algorithms (Chapters 5 and 8). In one case (Chapter 7), we experimented with freshly introduced concepts on real-world data. The study of graph envy-freeness (Chapter 4) is the only exception, as we did not provide any experimental study because of unavailability of suitable data. As opposed to other considered models, for graph envy-freeness neither real-world data nor established models for generating such data are available. Furthermore, the inputs of the related problems are quite complex. They consist of a social network and a list of resources together with the preferences of the agents over the resources. Thus, coming up from scratch with a reasonable model for such a complex input structure would require a very detailed and elaborate study by itself.

In Part I of the thesis, we studied problems related to fairly and efficiently allocating a collection of indivisible resources to a set of agents. We focused on identifying particular features of reality whose exploitation lead to computationally efficient solutions. However, for each topic in Part I the identified features served us in a different manner. Introducing graph envy-freeness in Chapter 4, we leveraged the fact that the classical notion of envy-freeness is too strong to properly reflect requirements occurring in the real world. We achieved this by incorporating information about social relations of the agents into the definition of envy-freeness. As a result we proposed a seemingly more complicated variant

of envy-freeness that allowed us to relax the classical notion of envy-freeness. Indeed, it is very unlikely in the real world that each agent compares itself to every other agent. Instead, according to the social comparison theory proposed by psychologist Leon Festinger [Fes54], agents rather compare themselves to their "socially-close" peers. We took advantage of this fact by considering only "local" comparisons between agents. Note that here the notion of locality is imposed by some domain-related measure such as relationship, knowledge of each other, physical proximity, or the like. In Chapter 5, we demonstrated quite a different way of utilizing real-world properties of fair allocation problems. Therein, we used two observations that led to further constraining the classical notion of envy-freeness. First, we used the fact that the number of agents might be rather limited in reality. Second, we exploited so-called bounded rationality [Bot+04]. This phenomenon roughly says that the preferences of human agents are not complicated (naturally, this does not apply to non-human agents). As shown in the thesis, we succeeded in both approaches of modifying the classical problem of finding envy-free and efficient allocations. Indeed, we were able to show a number of polynomial-time algorithms and fixed-parameter tractability results. Thus, our results in Part I form the following takeaway: Once looking for islands of tractability, try both restricting and relaxing the problem under investigation.

In Part II of the thesis, we focused on multiwinner voting problems emerging from intentional and unintentional changes in the voter preferences. Precisely, intentional changes were subject of our study in Chapter 8, while unintentional changes were considered in Chapter 7. For both the Robustness Radius and the Coalitional Manipulation problems we analyzed what changes in the voter preferences result in changing the outcome of an election. There are, however, two important differences between the two problems: who tried to alter an election outcome and what is the precise goal to achieve. In Robustness Radius, an organizer of an election was trying to find the smallest number of votes alterations that ensure that the outcome is changed—in any way. This is indeed different for Coalitional Manipulation, where a given set of manipulators aims at changing the outcome to achieve the one that satisfies them the most. Thus, compared to the model of Robustness Radius, in Coalitional Manipulation the manipulators not only had to change the votes properly but also had to agree on what outcome satisfies them the most. So, intuitively, studying how to achieve a satisfying manipulation in the Coalitional Manipulation model should be harder than studying how to change the election outcome in some arbitrary way in the Robustness

Radius problem. Indeed, our theoretical results from Part II reflect this intuition for the case of k-Approval. While computing the robustness radius for k-Approval is a computationally simple task, it is generally NP-hard to even decide whether a group of manipulators can successfully manipulate an election. Furthermore, our studies intuitively show that the general NP-hardness of Coalitional Manipulation comes mainly from the hardness of deciding which achievable outcome satisfies the manipulators the most (Proposition 8.13). Remarkably, Coalitional Manipulation is polynomial-time solvable for most cases for which the manipulators can easily agree on the most satisfying outcome (see Table 8.1). Summarizing, our theoretical findings in Part II seem to say that for a group of people agreeing on a common goal can be significantly harder than achieving this goal itself.

In the next section, we present a desirable example application of the topics studied in the thesis. By the example we illustrate several follow-up studies to our work, addressing further challenges related to fair allocation, multiwinner voting and, related issues.

9.1 Catching up with the Future

Coping with the SARS-CoV-2 pandemic, a vast majority of European countries faces severe issues related to resource management and resource distribution. This was especially evident in the first days of the outbreak when skyrocketing demand frequently caused acute shortages of necessary equipment. Just to mention few examples, lack of masks, ventilators, coronavirus tests, hand sanitizers, and human resources had been repeatedly reported by numerous institutions including hospitals, drug stores, firms and administrative units. Could computational social choice and, in particular, the issues studied in the thesis serve us in dealing with such cases?

Let us break down the aforementioned highly complex problem to address the posed question. For the discussion, we narrow down our focus only to managing different coronavirus tests and ventilators. We also assume that the supply of these is very limited and far below the demand, which indeed was the case in the early days of the pandemic. Thus, we neglect the supply at all and we arrive at having a (fixed) collection of indivisible resources. Assuming that managing hospitals is among the tasks of local governments (as, for example, in Poland), we represent units of administrative divisions (counties) as agents. Finally, asking the agents to report on their needs, we collect the data that form the preferences of the agents over different available tests and ventilators; for

example, a county could prefer a particular test over another because the former is quicker and thus more suitable to perform quick massive testing in some region (such massive testing was conducted for example in Silesia in Poland, where mines were identified to be hotspots for Covid-19 spread). And now the important question emerges: "How to properly allocate the resources?". Of course, we aim at allocating them in an efficient, non-wasteful manner, at the same time ensuring that all agents are treated fairly and thus feel well taken care of. Furthermore, we would like to be as robust as possible against unavoidable errors in reporting.

Despite the simplifications that we make, properly allocating the available resources involves considering all concepts that we studied. For example, we requested our allocation to be robust against possibly inaccurate data. Furthermore, to increase utilization of ventilators, we probably should consider "grouping" some counties and allocate them a shared bundle of ventilators. This way, the grouped counties can further distribute them "on demand" on a smaller and easier to manage scale. Additionally, it might be easier to aggregate preferences of all counties in a single group to decrease the amount and complexity of the collected data. Moreover, shrinking the data would certainly reduce the effort required to process the data in order to facilitate critical resources management. Last but not least, it is natural that only some counties may be grouped together. For example, counties that are far away from each other would not be able to manage ventilators well. Below, we briefly discuss the raised issues. In the discussion, we demonstrate possible high-level connections and interactions between the topics that we studied in the thesis.

Robustness In our pandemic scenario, the preferences of the agents could be susceptible to errors, perturbations and misreporting. It holds for various reasons: people tend to be vulnerable to making small mistakes, the data itself can be noisy, or agents could try to behave strategically. Hence, to achieve an allocation that is resistant to possible errors, it seems natural to study robustness in the context of fair allocation. The hope is that robust solutions would defend us, at least to some extent, against possibly costly and time-consuming changes of allocations. Studying the robustness of fair allocations was very recently initiated by Menon and Larson [ML20]. Yet, they only focused on the case of three agents, so many settings remain unstudied. Issues related to robustness have also gained attention in several other areas such as in the contexts of stable marriage [CSS19] and facility location [ML19].

Sharing and Grouping Sharing frequently leads to more efficient utilization of resources. The same rule applies to the case of ventilators in our scenario. Hence, it might be beneficial to let some counties share a pool of ventilators and use them on demand. This would prevent unnecessary idling which is extremely undesirable in our outbreak scenario featuring very high demand but no supply.

Incorporating sharing poses a big challenge to the models considered in the thesis. While, in the fair allocation scenario we assume private bundles and in multiwinner voting we deal with a single, collective set of resources. There is, however, a gap in modeling scenarios for indivisible resources between these two extreme models. We are aware of two works Manurangsi and Suksompong [MS17] and Sandomirskiy and Segal-Halevi [SS19] considering scenarios where agents can share the resources that are allocated to them. However, in both of them sharing the resources decreases the utility that is obtained by the agents who share the resource, as compared to the utility that an agent gets when the resource is not shared. This assumption contradicts our motivation, in which sharing can increase utilization of a resource by decreasing its idling time.

To cover our idea of shareable (ventilators) and non-shareable (tests) resources, we should probably apply lessons learned from both parts of the thesis. Even more so, as we aim at achieving a model that allows for efficiently finding desired allocations. Drawing from our conclusions from Part I, we should rather focus on structuring agent relations in the input and simplify agent preferences. Structuring agent relation appears relatively straightforward. For example, it is clear that distant counties will not be able to share ventilators. Simplifying agent preferences, however, requires more involved discussion. A relatively simple way to achieve this is to force counties to already provide simple preferences. We can do this by allowing them to choose only among such two options for each resources: high demand and no demand. Yet, this still may lead to harsh conflicts appearing within a single group that want to share ventilators (see Table 8.1, which features the NP-hardness of breaking ties even for this kind of utilities). Thus, to avoid the conflicts, it is probably beneficial to utilize our observation from Part II and first develop clear, joint goals for groups that want to share ventilators. We also expect the goals to be robust against small changes in the demand reported by the counties. Otherwise, it is questionable whether we can call the goals clear because a small perturbation in the demand could invalidate the goals. One way of obtaining such high-quality goals might be to run small multiwinner election within each group, treating the resources as candidates and the counties as voters. Then, we should check the robustness of the outcomes that the elections provide. If the outcomes turn

out to be robust, then they can be used as the clear goals of the groups and can be reported as the preference of the whole groups over the desired resources. Otherwise, the non-robust outcomes should be discussed more in the process of developing the goals. All in all, we believe that our insights from the thesis could indeed be useful to fill the gap between studying allocations of private bundles and selecting collective sets of resources.

9.2 Epilogue

The discussion provided in the above section is just the tip of the iceberg and a proper treatment of complex scenarios like this definitely requires very broad, interdisciplinary investigations. In particular, it shows a strong need for studying different kind of connections of multiwinner voting and fair allocation with other areas of computer science. For example, to cover the dynamics of the agents it would be beneficial to employ algorithmic game theory or theory of distributed systems.

Hopefully, one day our understanding of fair allocation, multiwinner voting, and their relations to other scientific disciplines will help us in dealing with such issues as in the depicted SARS-CoV-2 scenario. Meanwhile, may we remember that *not all those who wander are lost* [Tol12] when striving for this goal.

Bibliography

[ABM18] G. Amanatidis, G. Birmpas, and V. Markakis. "Comparing approximate relaxations of envy-freeness". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI '18)*. 2018, pp. 42–48 (cited on p. 68).

[Aca19] Academy of Motion Picture Arts and Sciences. *93rd oscars rules*. https://www.oscars.org/sites/oscars/files/93aa_rules.pdf. Accessed: 2020-08-10. 2019 (cited on p. 106).

[AKP17] R. Abebe, J. Kleinberg, and D. C. Parkes. "Fair division via social comparison". In: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '17)*. 2017, pp. 281–289 (cited on p. 19).

[AS20] H. Aziz and N. Shah. *Participatory budgeting: models and approaches*. 2020. arXiv: 2003.00606 [cs.GT] (cited on p. 4).

[AW19] M. Aleksandrov and T. Walsh. *Greedy algorithms for fair division of mixed manna*. 2019. arXiv: 1911.11005 [cs.AI] (cited on pp. 13, 101).

[Azi+17] H. Aziz, E. Elkind, P. Faliszewski, M. Lackner, and P. Skowron. "The Condorcet principle for multiwinner elections: From shortlisting to proportionality". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI '17)*. 2017, pp. 84–90 (cited on pp. 116, 120).

[Azi+18] H. Aziz, S. Bouveret, I. Caragiannis, I. Giagkousi, and J. Lang. "Knowledge, fairness, and social constraints". In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI '18)*. 2018, pp. 4638–4645 (cited on pp. 20, 28).

[Azi+19] H. Aziz, I. Caragiannis, A. Igarashi, and T. Walsh. "Fair allocation of indivisible goods and chores". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI '19)*. 2019, pp. 53–59 (cited on pp. 13, 89, 90, 92, 93, 101).

[BBN16] B. Bliem, R. Bredereck, and R. Niedermeier. "Complexity of efficient and envy-free resource allocation: few agents, resources, or utility levels". In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI' 16)*. 2016, pp. 102–108 (cited on pp. 13, 37, 38, 69, 78, 79).

[BC08] S. Barberà and D. Coelho. "How to choose a non-controversial list with k names". In: *Social Choice and Welfare* 31(1) (2008), pp. 79–96 (cited on p. 155).

[BC10] S. Barberà and D. Coelho. "On the rule of k names". In: *Games and Economic Behavior* 70(1) (2010), pp. 44–61 (cited on p. 120).

[BCM16] S. Bouveret, Y. Chevaleyre, and N. Maudet. "Fair allocation of indivisible goods". In: *Handbook of Computational Social Choice*. Ed. by F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. Cambridge University Press, 2016. Chap. 12, pp. 311–329 (cited on p. 13).

[Bei+21] X. Bei, A. Igarashi, X. Lu, and W. Suksompong. "The price of connectivity in fair division". In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*. To appear. 2021 (cited on p. 21).

[Ben+19] M. Bentert, J. Chen, V. Froese, and G. J. Woeginger. *Good things come to those who swap objects on paths*. 2019. arXiv: 1905.04219 [cs.DS] (cited on p. 21).

[Ben+21] M. Bentert, R. Bredereck, P. Györgyi, A. Kaczmarczyk, and R. Niedermeier. "A multivariate complexity analysis of the material consumption scheduling problem". In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*. To appear. 2021 (cited on p. ix).

[Bey+19] A. Beynier, Y. Chevaleyre, L. Gourvès, A. Harutyunyan, J. Lesca, N. Maudet, and A. Wilczynski. "Local envy-freeness in house allocation problems". In: *Autonomous Agents and Multi-Agents Systems* 33(5) (2019), pp. 591–627 (cited on p. 20).

[BFK20] R. Bredereck, T. Fluschnik, and A. Kaczmarczyk. *Multistage committee election*. 2020. arXiv: 2005.02300 [cs.GT] (cited on p. ix).

[BHS20] X. Bei, G. Huzhang, and W. Suksompong. "On the complexity of achieving proportional representation". In: *Social Choice and Welfare* 55(3) (2020), pp. 523–545 (cited on p. 25).

[Bil+19] V. Bilò, I. Caragiannis, M. Flammini, A. Igarashi, G. Monaco, D. Peters, C. Vinci, and W. S. Zwicker. "Almost envy-free allocations with connected bundles". In: *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ICTS '19)*. Vol. 124. 2019, 14:1–14:21 (cited on p. 21).

[BKN17] R. Bredereck, A. Kaczmarczyk, and R. Niedermeier. "On coalitional manipulation for multiwinner elections: shortlisting". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI '17)*. 2017, pp. 887–893 (cited on p. viii).

[BKN18] R. Bredereck, A. Kaczmarczyk, and R. Niedermeier. "Envy-free allocations respecting social networks". In: *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '18)*. 2018, pp. 283–291 (cited on pp. vii, 65).

[BKN20] R. Bredereck, A. Kaczmarczyk, and R. Niedermeier. "Electing successive committees: complexity and algorithms". In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*. 2020, pp. 1846–1853 (cited on p. ix).

[BKN21] R. Bredereck, A. Kaczmarczyk, and R. Niedermeier. "On coalitional manipulation for multiwinner elections: shortlisting". In: *Autonomous Agents and Multi-Agents Systems* 35 (2021), Article 38 (cited on pp. viii, 176, 177, 189, 197).

[BKV18] S. Barman, S. K. Krishnamurthy, and R. Vaish. "Finding fair and efficient allocations". In: *Proceedings of the 19th ACM Conference on Economics and Computation (EC '18)*. 2018, pp. 557–574 (cited on pp. 13, 89, 92).

[BL08] S. Bouveret and J. Lang. "Efficiency and envy-freeness in fair division of indivisible goods: logical representation and complexity". In: *Journal of Artificial Intelligence Research* 32(1) (2008), pp. 525–564 (cited on pp. 13, 37, 78, 97).

[BL16] S. Bouveret and M. Lemaître. "Characterizing conflicts in fair division of indivisible goods using a scale of criteria". In: *Autonomous Agents and Multi-Agents Systems* 30(2) (2016), pp. 259–290 (cited on p. 20).

[BNW11] N. Betzler, R. Niedermeier, and G. J. Woeginger. "Unweighted coalitional manipulation under the Borda rule is NP-hard". In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*. 2011, pp. 55–60 (cited on p. 155).

[Boe+20] N. Boehmer, R. Bredereck, P. Faliszewski, A. Kaczmarczyk, and R. Niedermeier. "Line-up elections: parallel voting with shared candidate pool". In: *Proceedings of the 13th International Symposium on Algorithmic Game Theory (SAGT '20)*. 2020, pp. 275–290 (cited on p. ix).

[Bot+04] W. Bottom, T. Gilovich, D. Griffin, and D. Kahneman. "Heuristics and biases: the psychology of intuitive judgment". In: *The Academy of Management Review* 29 (2004), p. 695 (cited on pp. 68, 208).

[Bou+17] S. Bouveret, K. Cechlárová, E. Elkind, A. Igarashi, and D. Peters. "Fair division of a graph". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI '17)*. 2017, pp. 135–141 (cited on p. 21).

[BQZ17] X. Bei, Y. Qiao, and S. Zhang. "Networked fairness in cake cutting". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI '17)*. 2017, pp. 3632–3638 (cited on p. 19).

[BR15] D. Baumeister and J. Rothe. "Preference aggregation by voting". In: *Economics and Computation: An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*. Ed. by J. Rothe. Springer, 2015. Chap. 4, pp. 197–325 (cited on pp. 108, 154, 157).

[BR16] C. Boutilier and J. S. Rosenschein. "Incomplete information and communication in voting". In: *Handbook of Computational Social Choice*. Ed. by F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. Cambridge University Press, 2016, pp. 223–257 (cited on p. 111).

[Bra+16] F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, eds. *Handbook of Computational Social Choice*. Cambridge University Press, 2016 (cited on p. 108).

[Bre+16] R. Bredereck, P. Faliszewski, R. Niedermeier, and N. Talmon. "Complexity of shift bribery in committee elections". In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI '16)*. Accepted for publication in the ACM Transactions on Computation Theory. 2016, pp. 2452–2458 (cited on p. 136).

[Bre+17] R. Bredereck, P. Faliszewski, A. Kaczmarczyk, R. Niedermeier, P. Skowron, and N. Talmon. "Robustness among multiwinner voting rules". In: *Proceedings of the 10th International Symposium on Algorithmic Game Theory (SAGT '17)*. 2017, pp. 80–92 (cited on p. viii).

[Bre+19a] R. Bredereck, P. Faliszewski, A. Kaczmarczyk, and R. Niedermeier. "An experimental view on committees providing justified representation". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI '19)*. 2019, pp. 109–115 (cited on pp. ix, 199).

[Bre+19b] R. Bredereck, A. Kaczmarczyk, D. Knop, and R. Niedermeier. "High-multiplicity fair allocation: lenstra empowered by N-fold integer programming". In: *Proceedings of the 2019 ACM Conference on Economics and Computation (EC '19)*. 2019, pp. 505–523 (cited on pp. viii, 89, 103).

[Bre+20a] R. Bredereck, P. Faliszewski, M. Furdyna, A. Kaczmarczyk, and M. Lackner. "Strategic campaign management in apportionment elections". In: *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*. 2020, pp. 103–109 (cited on p. ix).

[Bre+20b] R. Bredereck, P. Faliszewski, A. Kaczmarczyk, D. Knop, and R. Niedermeier. "Parameterized algorithms for finding a collective set of items". In: *The 34th AAAI Conference on Artificial Intelligence (AAAI '20*. 2020, pp. 1838–1845 (cited on pp. ix, 4).

[Bre+20c] R. Bredereck, A. Kaczmarczyk, D. Knop, and R. Niedermeier. *High-multiplicity fair allocation using parametric integer linear programming*. 2020. arXiv: 2005.04907 [cs.GT] (cited on p. ix).

[Bre+21a] R. Bredereck, P. Faliszewski, A. Kaczmarczyk, R. Niedermeier, P. Skowron, and N. Talmon. "Robustness among multiwinner voting rules". In: *Artificial Intelligence* 290 (2021), 103403 (cited on pp. viii, 148, 150).

[Bre+21b] R. Bredereck, A. Figiel, A. Kaczmarczyk, D. Knop, and R. Niedermeier. "High-multiplicity fair allocation made more practical". In: *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '21)*. 2021, pp. 260–268 (cited on p. ix).

[BST19] M. Blom, P. J. Stuckey, and V. Teague. "Toward computing the margin of victory in single transferable vote elections". In: *INFORMS Journal on Computing* 31(4) (2019), pp. 636–653 (cited on pp. 114, 122).

[BSU13] N. Betzler, A. Slinko, and J. Uhlmann. "On the computation of fully proportional representation". In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 475–519 (cited on pp. 116, 117).

[BSZ91] S. Barberà, H. Sonnenschein, and L. Zhou. "Voting by committees". In: *Econometrica* 59(3) (1991), pp. 595–609 (cited on p. 108).

[BT96] S. J. Brams and A. D. Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996 (cited on p. 13).

[BTT89] J. J. Bartholdi III, C. A. Tovey, and M. A. Trick. "The computational difficulty of manipulating an election". In: *Social Choice and Welfare* 6(3) (1989), pp. 227–241 (cited on p. 155).

[Bur] Bureau of Labor Statistics, U.S. Department of Labor. *Employer costs for employee compensation—June 2017*. https://www.bls.gov/news.release/pdf/ecec.pdf. Accessed 2020-08-10 (cited on p. 17).

[Car+16] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang. "The unreasonable fairness of maximum Nash welfare". In: *Proceedings of the 17th ACM Conference on Economics and Computation (EC '16)*. 2016, pp. 305–322 (cited on pp. 68, 89, 90, 92).

[Car11] D. Cary. "Estimating the margin of victory for instant-runoff voting". Presented at 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections. 2011 (cited on pp. 114, 122).

[CC83] J. Chamberlin and P. Courant. "Representative deliberations and representative decisions: Proportional representation and the Borda rule". In: *American Political Science Review* 77(3) (1983), pp. 718–733 (cited on p. 115).

[CCC06] L. Cai, S. M. Chan, and S. O. Chan. "Random separation: a new method for solving fixed-cardinality optimization problems". In: *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC '06)*. 2006, pp. 239–250 (cited on p. 50).

[CEM17] Y. Chevaleyre, U. Endriss, and N. Maudet. "Distributed fair allocation of indivisible goods". In: *Artificial Intelligence* 242 (2017), pp. 1–22 (cited on pp. 21, 22, 62).

[CGM20] B. R. Chaudhury, J. Garg, and K. Mehlhorn. "EFX exists for three agents". In: *Proceedings of the 21st ACM Conference on Economics and Computation (EC '20)*. 2020, pp. 1–19 (cited on p. 92).

[Cha+19] A. Chakraborty, G. K. Patro, N. Ganguly, K. P. Gummadi, and P. Loiseau. "Equality of voice: towards fair representation in crowdsourced Top-K recommendations". In: *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT '19)*. 2019, pp. 129–138 (cited on p. 1).

[Coe04] D. Coelho. "Understanding, Evaluating and Selecting Voting Rules Through Games and Axioms". PhD thesis. Universitat Autònoma de Barcelona, 2004 (cited on p. 118).

[CRX09] V. Conitzer, M. Rognlie, and L. Xia. "Preference functions that score rankings and maximum likelihood estimation". In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*. 2009, pp. 109–115 (cited on pp. 116, 122, 136, 149).

[CS05] V. Conitzer and T. Sandholm. "Common voting rules as maximum likelihood estimators". In: *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence (UAI '05)*. 2005, pp. 145–152 (cited on p. 149).

[CSL07] V. Conitzer, T. Sandholm, and J. Lang. "When are elections with few candidates hard to manipulate?" In: *Journal of the ACM* 54(3) (2007), pp. 1–33 (cited on pp. 155, 203).

[CSS19] J. Chen, P. Skowron, and M. Sorge. "Matchings under preferences: strength of stability and trade-offs". In: *Proceedings of the 2019 ACM Conference on Economics and Computation (EC '19)*. 2019, pp. 41–59 (cited on p. 210).

[CT07] T. Coleman and V. Teague. "On the complexity of manipulating elections." In: *Proceedings of Computing: The 13th Australasian Theory Symposium*. 2007, pp. 25–33 (cited on p. 155).

[CW16] V. Conitzer and T. Walsh. "Barriers to manipulation in voting". In: *Handbook of Computational Social Choice*. Ed. by F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. Cambridge University Press, 2016. Chap. 6, pp. 126–145 (cited on pp. 154, 157, 205).

[Cyg+15] M. Cygan, F. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015 (cited on pp. 8, 130).

[Dav+14] J. Davies, G. Katsirelos, N. Narodytska, T. Walsh, and L. Xia. "Complexity of and algorithms for the manipulation of Borda, Nanson's and Baldwin's voting rules". In: *Artificial Intelligence* 217 (2014), pp. 20–42 (cited on p. 155).

[DD16] M. Diss and A. Doghmi. "Multi-winner scoring election methods: Condorcet consistency and paradoxes". In: *SSRN Electronic Journal* (2016), pp. 97–116 (cited on p. 110).

[Deb92] B. Debord. "An axiomatic characterization of Borda's k-choice function". In: *Social Choice and Welfare* 9(4) (1992), pp. 337–343 (cited on p. 158).

[DF13] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013 (cited on p. 8).

[DF95] R. G. Downey and M. R. Fellows. "Fixed-parameter tractability and completeness ii: on completeness for W[1]". In: *Theoretical Computer Science* 141(1) (1995), pp. 109–131 (cited on p. 8).

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999 (cited on p. 173).

[DN15] P. Dey and Y. Narahari. "Estimating the margin of victory of an election using sampling". In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI '15)*. 2015, pp. 1120–1126 (cited on p. 122).

[Dro81] H. R. Droop. "On methods of electing representatives". In: *Journal of the Statistical Society of London* 44(2) (1881), pp. 141–202 (cited on p. 120).

[DS12] B. Dorn and I. Schlotter. "Multivariate complexity analysis of swap bribery". In: *Algorithmica* 64(1) (2012), pp. 126–151 (cited on p. 137).

[Edu+20] E. Eduard, G. Robert, H. Thekla, and O. Sebastian. "Parameterized complexity of envy-free resource allocations in social networks". In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*. 2020, pp. 7135–7142 (cited on pp. 20, 62, 65).

[EFS09] E. Elkind, P. Faliszewski, and A. Slinko. "Swap bribery". In: *Proceedings of the 2nd International Symposium on Algorithmic Game Theory (SAGT '09)*. 2009, pp. 299–310 (cited on pp. 114, 122).

[EHK18] F. Eisenbrand, C. Hunkenschröder, and K.-M. Klein. "Faster Algorithms for Integer Programs with Block Structure". In: *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP '18)*. Vol. 107. 2018, 49:1–49:13 (cited on pp. 76, 77).

[Elk+17] E. Elkind, P. Faliszewski, P. Skowron, and A. M. Slinko. "Properties of multiwinner voting rules". In: *Social Choice and Welfare* 48(3) (2017), pp. 599–632 (cited on pp. 106, 111, 115, 154, 155).

[End17] U. Endriss, ed. *Trends in Computational Social Choice*. AI Access, 2017 (cited on p. 108).

[Erd+15] G. Erdélyi, M. R. Fellows, J. Rothe, and L. Schend. "Control complexity in Bucklin and fallback voting: an experimental analysis". In: *Journal of Computer and System Sciences* 81(4) (2015), pp. 661–670 (cited on p. 155).

[ES16] E. Elkind and A. Slinko. "Rationalizations of voting rules". In: *Handbook of Computational Social Choice*. Ed. by F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. Cambridge University Press, 2016. Chap. 8 (cited on p. 148).

[Fal+17] P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. "Multiwinner voting: a new challenge for social choice theory". In: *Trends in Computational Social Choice*. Ed. by U. Endriss. AI Access Foundation, 2017, pp. 27–47 (cited on p. 106).

[Fal+18a] P. Faliszewski, M. Lackner, D. Peters, and N. Talmon. "Effective heuristics for committee scoring rules". In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI '18)*. 2018, pp. 1023–1030 (cited on p. 199).

[Fal+18b] P. Faliszewski, A. Slinko, K. Stahl, and N. Talmon. "Achieving fully proportional representation by clustering voters". In: *Journal of Heuristics* 24(5) (2018), pp. 725–756 (cited on p. 117).

[Fal+19] P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. "Committee scoring rules: Axiomatic characterization and hierarchy". In: *ACM Transactions on Economics and Computation* 6(1) (2019), Article 3 (cited on pp. 106, 111, 115).

[Fel+09] M. R. Fellows, D. Hermelin, F. Rosamond, and S. Vialette. "On the parameterized complexity of multiple-interval graph problems". In: *Theoretical Computer Science* 410(1) (2009), pp. 53–61 (cited on p. 174).

[Fes54] L. Festinger. "A theory of social comparison processes". In: *Human Relations* 7(2) (1954), pp. 117–140 (cited on pp. 18, 208).

[FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006 (cited on p. 8).

[FR15] P. Faliszewski and J. Rothe. "Control and bribery in voting". In: *Handbook of Computational Social Choice*. Ed. by F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. Cambridge University Press, 2015. Chap. 7 (cited on p. 106).

[FT17] A. Filtser and N. Talmon. “Distributed monitoring of election winners”. In: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS ’17)*. 2017, pp. 1160–1168 (cited on p. 122).

[FT87] A. Frank and É. Tardos. “An application of simultaneous diophantine approximation in combinatorial optimization”. In: *Combinatorica* 7(1) (1987), pp. 49–65 (cited on p. 11).

[Geh85] W. Gehrlein. “The Condorcet criterion and committee selection”. In: *Mathematical Social Sciences* 10(3) (1985), pp. 199–209 (cited on p. 118).

[GF19] G. Gawron and P. Faliszewski. “Robustness of approval-based multiwinner voting rules”. In: *Proceedings of the 6th International Conference on Algorithmic Decision Theory (SAGT ’19)*. 2019, pp. 17–31 (cited on pp. 115, 150).

[GF81] W. V. Gehrlein and P. C. Fishburn. “Constant scoring rules for choosing one among many alternatives”. In: *Quality and Quantity* 15 (1981), pp. 203–210 (cited on p. 110).

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979 (cited on p. 126).

[Glo20] Global Media & Entertainment Ltd. *Classic FM hall of fame 2020*. https://halloffame.classicfm.com/2020. Accessed: 2020-08-10. 2020 (cited on p. 154).

[GLS81] M. Grötschel, L. Lovász, and A. Schrijver. “The ellipsoid method and its consequences in combinatorial optimization”. In: *Combinatorica* 1(2) (1981), pp. 169–197 (cited on p. 97).

[GLW17] L. Gourvès, J. Lesca, and A. Wilczynski. “Object allocation via swaps along a social network”. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI ’17*. 2017, pp. 213–219 (cited on p. 21).

[GMT18] L. Gourvès, J. Monnot, and L. Tlilane. “Subset sum problems with digraph constraints”. In: *Journal of Combinatorial Optimization* 36(3) (2018), pp. 937–964 (cited on p. 21).

[Gon85] T. F. Gonzalez. “Clustering to minimize the maximum intercluster distance”. In: *Theoretical Computer Science* 38 (1985), pp. 293–306 (cited on p. 126).

[GP15] J. Goldman and A. D. Procaccia. “Spliddit: unleashing fair division algorithms”. In: *SIGecom Exchanges* 13(2) (2015), pp. 41–46 (cited on pp. 3, 98).

[Gra75] J. E. Graver. "On the foundations of linear and integer linear programming I". In: *Mathematical Programming* 9(1) (1975), pp. 207–226 (cited on pp. 75, 76).

[Guo+16] C. Guo, Y. Zhang, M. Sheng, X. Wang, and Y. Li. "α-fair power allocation in spectrum-sharing networks". In: *IEEE Transactions on Vehicular Technology* 65(5) (2016), pp. 3771–3777 (cited on p. 1).

[Hai+15] M. Haider, A. Aamir, A. Hamid, and M. Hashim. "A literature analysis on the importance of non-financial rewards for employees' job satisfaction". In: *Abasyn University Journal of Social Sciences* 8 (2015), pp. 341–354 (cited on p. 18).

[HKW10] R. Hemmecke, M. Köppe, and R. Weismantel. "A polynomial-time algorithm for optimizing over N-fold 4-block decomposable integer programs". In: *Proceedings of the 14th International Conference on Integer Programming and Combinatorial Optimization (IPCO '10)*. 2010, pp. 219–229 (cited on pp. 67, 75).

[HOR13] R. Hemmecke, S. Onn, and L. Romanchuk. "N-fold integer programming in cubic time". In: *Mathematical Programming* 137(1–2) (2013), pp. 325–341 (cited on pp. 67, 75, 76).

[HS07] S. Hoşten and S. Sullivant. "A finiteness theorem for Markov bases of hierarchical models". In: *Journal of Combinatorial Theory, Series A* 114(2) (2007), pp. 311–321 (cited on p. 76).

[Ide20] Ideal. *Shortlisting step-by-step guide for candidate recruitment*. https://ideal.com/shortlisting/. Accessed: 2020-08-08. 2020 (cited on pp. 106, 154).

[IP19] A. Igarashi and D. Peters. "Pareto-optimal allocation of indivisible goods with connectivity constraints". In: *The 33rd AAAI Conference on Artificial Intelligence (AAAI '19)*. 2019, pp. 2045–2052 (cited on p. 21).

[Jan+13] K. Jansen, S. Kratsch, D. Marx, and I. Schlotter. "Bin packing with fixed number of bins revisited". In: *Journal of Computer and System Sciences* 79(1) (2013), pp. 39–49 (cited on pp. 53, 61).

[Kal19] L. Kalkbrenner. "Coalitional Manipulation for Multiwinner Elections: Algorithms and Experiments". Bachelor Thesis. TU Berlin, 2019 (cited on p. 199).

[Kam03] T. Kamishima. "Nantonac collaborative filtering: Recommendation based on order responses". In: *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD '03)*. 2003, pp. 583–588 (cited on pp. 146, 148).

[Kan87] R. Kannan. “Minkowski’s convex body theorem and integer programming”. In: *Mathematics of Operations Research* 12(3) (1987), pp. 415–440 (cited on p. 11).

[Kei+09] B. de Keijzer, S. Bouveret, T. Klos, and Y. Zhang. “On the complexity of efficiency and envy-freeness in fair division of indivisible goods with additive preferences”. In: *Proceedings of the 1st International Conference on Algorithmic Decision Theory (ADT ’09)*. 2009, pp. 98–110 (cited on p. 13).

[Ken38] M. G. Kendall. “A new Measure of Rank Correlation”. In: *Biometrika* 30(1–2) (1938), pp. 81–93 (cited on p. 112).

[KF19] A. Kaczmarczyk and P. Faliszewski. “Algorithms for destructive shift bribery”. In: *Autonomous Agents and Multi-Agent Systems* 33(3) (2019), pp. 275–297 (cited on pp. ix, 122, 129, 146).

[KKM18] D. Knop, M. Koutecký, and M. Mnich. “A unifying framework for manipulation problems”. In: *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS ’18)*. 2018, pp. 256–264 (cited on p. 72).

[KKM20a] D. Knop, M. Koutecký, and M. Mnich. “Combinatorial n-fold integer programming and applications”. In: *Mathematical Programming* 184(1) (2020), pp. 1–34 (cited on pp. 75, 80).

[KKM20b] D. Knop, M. Koutecký, and M. Mnich. “Voting and bribing in single-exponential time”. In: *ACM Transactions on Economics and Computation* 8(3) (2020), 12:1–12:28 (cited on p. 137).

[KM15] E. Kamwa and V. Merlin. “Scoring rules over subsets of alternatives: consistency and paradoxes”. In: *Journal of Mathematical Economics* 61 (2015), pp. 130–138 (cited on p. 110).

[KPP04] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004 (cited on pp. 179, 182).

[LB11] T. Lu and C. Boutilier. “Budgeted social choice: From consensus to personalized decision making”. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI ’11)*. 2011, pp. 280–286 (cited on pp. 116, 117).

[Len83] H. W. Lenstra. “Integer programming with a fixed number of variables”. In: *Mathematics of Operations Research* 8(4) (1983), pp. 538–548 (cited on pp. 11, 36, 54, 55, 67, 80, 86, 137).

[LHK13] J. A. D. Loera, R. Hemmecke, and M. Köppe. *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*. Vol. 14. MOS-SIAM Series on Optimization. 2013 (cited on p. 75).

[Lin11] A. Lin. "The complexity of manipulating k-approval elections". In: *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence (ICAART '11)*. 2011, pp. 212–218 (cited on pp. 157, 203).

[Lip+04] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. "On approximately fair allocations of indivisible goods". In: *Proceedings of the 5th ACM Conference on Electronic Commerce (EC '04)*. 2004, pp. 125–131 (cited on pp. 68, 89, 92).

[LR19] P. Lange and J. Rothe. "Optimizing social welfare in social networks". In: *Proceedings of 6th International Conference on Algorithmic Decision Theory (ADT '19)*. 2019, pp. 81–96 (cited on p. 62).

[Lu+12] T. Lu, P. Tang, A. D. Procaccia, and C. Boutilier. "Bayesian vote manipulation: Optimal strategies and impact on welfare". In: *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI '12)*. 2012, pp. 543–553 (cited on p. 155).

[LX15] J. Lang and L. Xia. "Voting in combinatorial domains". In: *Handbook of Computational Social Choice*. Ed. by F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. Cambridge University Press, 2015. Chap. 9 (cited on p. 106).

[Mag+11] T. Magrino, R. Rivest, E. Shen, and D. Wagner. "Computing the margin of victory in IRV elections". Presented at 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections. 2011 (cited on pp. 114, 122).

[Mar02] T. Martino. "Fair chore division for climate change". In: *Social Theory and Practice* 28(1) (2002), pp. 101–134 (cited on p. 1).

[Mar17] E. Markakis. "Approximation algorithms and hardness results for fair division with indivisible good". In: *Trends in Computational Social Choice*. Ed. by U. Endriss. AI Access, 2017. Chap. 12, pp. 231–247 (cited on p. 13).

[Mei+08] R. Meir, A. D. Procaccia, J. S. Rosenschein, and A. Zohar. "Complexity of strategic behavior in multi-winner elections". In: *Journal of Artificial Intelligence Research* 33(1) (2008), pp. 149–178 (cited on pp. 110, 155–159, 163, 203).

[Min19] Ministry of Science and Higher Education of the Republic of Poland. *Informations on the election of The Board of Research Excellence (in Polish)*. http://www.bip.nauka.gov.pl/g2/oryginal/2019_03/c435c5061f0aab7158eba2716553f240.pdf. Accessed: 2020-08-07. 2019 (cited on pp. 1, 155).

[ML19] V. Menon and K. Larson. “Mechanism design for locating a facility under partial information”. In: *Proceedings of the 12th International Symposium on Algorithmic Game Theory (SAGT ’19)*. Ed. by D. Fotakis and E. Markakis. 2019, pp. 49–62 (cited on p. 210).

[ML20] V. Menon and K. Larson. *Algorithmic stability in fair allocation of indivisible goods among two agents*. 2020. arXiv: 2007.15203 [cs.GT] (cited on p. 210).

[Mou03] H. Moulin. *Fair Division and Collective Welfare*. MIT Press, 2003 (cited on p. 13).

[MP13] D. Marx and M. Pilipczuk. *Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask)*. 2013. arXiv: 1307.2187 [cs.CC] (cited on pp. 46, 47).

[MP14] D. Marx and M. Pilipczuk. “Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism (but were afraid to ask)”. In: *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS ’14)*. 2014, pp. 542–553 (cited on p. 46).

[MS17] P. Manurangsi and W. Suksompong. “Asymptotic existence of fair divisions for groups”. In: *Mathematical Social Sciences* 89 (2017), pp. 100–108 (cited on p. 211).

[MS19] N. Misra and C. Sonar. “Robustness radius for Chamberlin-Courant on restricted domains”. In: *Proceedings of the 45th International Conference on Current Trends in Theory and Practice of Computer Science*. 2019, pp. 341–353 (cited on pp. 114, 150).

[MW13] N. Mattei and T. Walsh. “Preflib: a library for preferences”. In: *Proceedings of the 3nd International Conference on Algorithmic Decision Theory (ADT ’13)*. 2013, pp. 259–270 (cited on pp. 3, 112, 114, 146).

[Nie06] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006 (cited on p. 8).

[Nis+07] N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007 (cited on p. 205).

[Onn10] S. Onn. “Nonlinear discrete optimization”. In: *Zurich Lectures in Advanced Mathematics, European Mathematical Society* (2010) (cited on pp. 75, 76).

[OZE13] S. Obraztsova, Y. Zick, and E. Elkind. “On manipulation in multi-winner elections based on scoring rules”. In: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS ’13)*. 2013, pp. 359–366 (cited on pp. 108, 157, 163, 203).

[Pap81] C. H. Papadimitriou. “On the complexity of integer programming”. In: *Journal of the ACM* 28(4) (1981), pp. 765–768 (cited on p. 81).

[Pet18] D. Peters. "Single-peakedness and total unimodularity: New polynomial-time algorithms for multi-winner elections". In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI '18)*. 2018, pp. 1169–1176 (cited on p. 117).

[PR18] B. Plaut and T. Roughgarden. "Almost envy-freeness with general valuations". In: *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '18)*. 2018, pp. 2584–2603 (cited on pp. 68, 89, 90).

[Pre29] M. Presburger. "Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt". In: *Sprawozdanie z 1 kongresu matematyków krajów słowiańskich. Comptes Rendus du I congrés de Mathématiciens des Pays Slaves*. 1929, pp. 92–101 (cited on p. 77).

[Pro+20] A. Procaccia, N. Shah, J. Goldman, and D. Kurokawa. *Fair Division of Rent, Goods, Credit, Fare, and Tasks - Spliddit*. http://www.spliddit.org/. Accessed: 2020-08-07. 2020 (cited on pp. 3, 67, 98).

[Pro13] A. D. Procaccia. "Cake cutting: Not just child's play". In: *Communications of the ACM* 56(7) (2013), pp. 78–87 (cited on p. 13).

[Pro16] A. D. Procaccia. "Cake cutting algorithms". In: *Handbook of Computational Social Choice*. Ed. by F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. Cambridge University Press, 2016. Chap. 13, pp. 311–329 (cited on p. 13).

[PRZ08] A. Procaccia, J. Rosenschein, and A. Zohar. "On the complexity of achieving proportional representation". In: *Social Choice and Welfare* 30(3) (2008), pp. 353–362 (cited on pp. 116, 117).

[PZ90] J. W. Pratt and R. J. Zeckhauser. "The fair and efficient division of the Windsor family silver". In: *Management Science* 36(11) (1990), pp. 1293–1301 (cited on p. 1).

[Rec20] Recording Academy. *Grammy awards voting process*. https://www.grammy.com/grammys/awards/voting-process. Accessed: 2020-08-07. 2020 (cited on p. 1).

[RW98] J. M. Robertson and W. A. Webb. *Cake-Cutting Algorithms: Be Fair if You Can*. A K Peters, 1998 (cited on p. 13).

[Sch+19] J. Scheuerman, J. L. Harman, N. Mattei, and K. B. Venable. *Heuristics in multi-winner approval voting*. 2019. arXiv: 1905.12104 [cs.GT] (cited on pp. 2, 155).

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., 1986 (cited on pp. 11, 97).

[SFL16] P. Skowron, P. Faliszewski, and J. L. Lang. “Finding a collective set of items: From proportional multirepresentation to group recommendation”. In: *Artificial Intelligence* 241 (2016), pp. 191–216 (cited on pp. 4, 147).

[SFS15] P. Skowron, P. Faliszewski, and A. Slinko. “Achieving fully proportional representation: Approximability results”. In: *Artificial Intelligence* 222 (2015), pp. 67–103 (cited on p. 117).

[SFS19] P. Skowron, P. Faliszewski, and A. Slinko. “Axiomatic characterization of committee scoring rules”. In: *Journal of Economic Theory* 180 (2019), pp. 244–273 (cited on pp. 111, 115).

[Sko+15] P. Skowron, L. Yu, P. Faliszewski, and E. Elkind. “The complexity of fully proportional representation for single-crossing electorates”. In: *Theoretical Computer Science* 569 (2015), pp. 43–57 (cited on p. 117).

[SS19] F. Sandomirskiy and E. Segal-Halevi. *Fair division with minimal sharing*. 2019. arXiv: 1908.01669 [cs.GT] (cited on pp. 105, 211).

[SSX17] S. Sekar, S. Sikdar, and L. Xia. “Condorcet consistent bundling with social choice”. In: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS ’17)*. 2017, pp. 33–41 (cited on p. 120).

[Ste48] H. Steinhaus. “The problem of fair division”. In: *Econometrica* 16 (1948), pp. 101–104 (cited on p. 13).

[Sti+10] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova. “Resource allocation algorithms for virtualized service hosting platforms”. In: *Journal of Parallel and Distributed Computing* 70(9) (2010), pp. 962–974 (cited on p. 1).

[Suk17] W. Suksompong. “Fairly allocating contiguous blocks of indivisible items”. In: *Proceedings of the 10th International Symposium on Algorithmic Game Theory (SAGT ’17)*. 2017, pp. 333–344 (cited on p. 21).

[SW18] A. Saffidine and A. Wilczynski. “Constrained swap dynamics over a social network in distributed resource reallocation”. In: *Proceedings of the 11th International Symposium on Algorithmic Game Theory (SAGT ’18)*. 2018, pp. 213–225 (cited on p. 21).

[SYE13] D. Shiryaev, L. Yu, and E. Elkind. “On elections with robust winners”. In: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS ’13)*. 2013, pp. 415–422 (cited on pp. 114, 122, 129).

[Szu+20] S. Szufa, P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. "Drawing a map of elections in the space of statistical cultures". In: *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '20)*. 2020, pp. 1341–1349 (cited on p. 200).

[Tol12] J. Tolkien. *The Fellowship of the Ring*. The Lord of the Rings. Orlando: Houghton Mifflin Harcourt, 2012 (cited on p. 212). Repr. of *The Fellowship of the Ring*. The Lord of the Rings. London: George Allen & Unwin, 1954.

[TR00] T. Tideman and D. Richardson. "Better voting methods through technology: the refinement-manageability trade-off in the single transferable vote". In: *Public Choice* 103 (2000), pp. 13–34 (cited on p. 139).

[Wal11] T. Walsh. "Where are the hard manipulation problems?" In: *Journal of Artificial Intelligence Research* 44 (2011), pp. 1–29 (cited on p. 155).

[WS98] D. J. Watts and S. H. Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393(6684) (1998), pp. 440–442 (cited on p. 28).

[Xia12] L. Xia. "Computing the margin of victory for various voting rules". In: *Proceedings of the 13th ACM Conference on Electronic Commerce (EC '12)*. 2012, pp. 982–999 (cited on pp. 114, 122).

Schriftenreihe **Foundations of computing**
Hrsg.: Prof. Dr. Stephan Kreutzer, Prof. Dr. Uwe Nestmann, Prof. Dr. Rolf Niedermeier

ISSN 2199-5249 (print)
ISSN 2199-5257 (online)

01: Bevern, René van: Fixed-Parameter Linear-Time Algorithms for NP-hard Graph and Hypergraph Problems Arising in Industrial Applications. - 2014. - 225 S.
ISBN **978-3-7983-2705-4** (print) EUR **12,00**
ISBN **978-3-7983-2706-1** (online)

02: Nichterlein, André: Degree-Constrained Editing of Small-Degree Graphs. - 2015. - xiv, 225 S.
ISBN **978-3-7983-2705-4** (print) EUR **12,00**
ISBN **978-3-7983-2706-1** (online)

03: Bredereck, Robert: Multivariate Complexity Analysis of Team Management Problems. - 2015. - xix, 228 S.
ISBN **978-3-7983-2764-1** (print) EUR **12,00**
ISBN **978-3-7983-2765-8** (online)

04: Talmon, Nimrod: Algorithmic Aspects of Manipulation and Anonymization in Social Choice and Social Networks. - 2016. - xiv, 275 S.
ISBN **978-3-7983-2804-4** (print) EUR **13,00**
ISBN **978-3-7983-2805-1** (online)

05: Siebertz, Sebastian: Nowhere Dense Classes of Graphs. Characterisations and Algorithmic Meta-Theorems. - 2016. - xxii, 149 S.
ISBN **978-3-7983-2818-1** (print) EUR **11,00**
ISBN **978-3-7983-2819-8** (online)

06: Chen, Jiehua: Exploiting Structure in Computationally Hard Voting Problems. - 2016. - xxi, 255 S.
ISBN **978-3-7983-2825-9** (print) EUR **13,00**
ISBN **978-3-7983-2826-6** (online)

07: Arbach, Youssef: On the Foundations of dynamic coalitions. Modeling changes and evolution of workflows in healthcare scenarios - 2016. - xv, 171 S.
ISBN **978-3-7983-2856-3** (print) EUR **12,00**
ISBN **978-3-7983-2857-0** (online)

08: Sorge, Manuel: Be sparse! Be dense! Be robust! Elements of parameterized algorithmmics. - 2017. - xvi, 251 S.
ISBN 978-3-7983-2885-3 (print) EUR **13,00**
ISBN 978-3-7983-2886-0 (online)

09: Dittmann, Christoph: Parity games, separations, and the modal μ-calculus. - 2017. - x, 274 S.
ISBN **978-3-7983-2887-7** (print) EUR **13,00**
ISBN **978-3-7983-2888-4** (online)

10: Karcher, David S.: Event Structures with Higher-Order Dynamics. - 2019. - xix, 125 S.
ISBN **978-3-7983-2995-9** (print) EUR **11,00**
ISBN **978-3-7983-2996-6** (online)

11: Jungnickel, Tim: On the Feasibility of Multi-Leader Replication in the Early Tiers. - 2018. - xiv, 177 S.
ISBN **978-3-7983-3001-6** (print) EUR **13,00**
ISBN **978-3-7983-3002-3** (online)

12: Froese, Vincent: Fine-grained complexity analysis of some combinatorial data science problems. - 2018. - xiv, 166 S.
ISBN **978-3-7983-3003-0** (print) EUR **11,00**
ISBN **978-3-7983-3004-7** (online)

13: Molter, Hendrik: Classic graph problems made temporal – a parameterized complexity analysis. - 2020. - xii, 206 S.
ISBN **978-3-7983-3172-3** (print) EUR **12,00**
ISBN **978-3-7983-3173-0** (online)

14: Bentert, Matthias: Elements of Dynamic and 2-SAT Programming: Paths, Trees, and Cuts. - 2021. - xiv, 199 S.
ISBN **978-3-7983-3209-6** (print)
ISBN **978-3-7983-3210-2** (online)

Universitätsverlag der TU Berlin

Algorithmic Aspects of Resource Allocation and Multiwinner Voting: Theory and Experiments

This thesis investigates elements of computational social choice in the light of real-world applications. We propose several new notions and extensions of existing models—among others, we augment the scenario of fair allocation of indivisible resources with the social context and investigate the robustness of multiwinner election outcomes. Then we analyze the complexity of answering the computational questions raised by the notions we propose.
Our theoretical study reveals that the introduced concepts lead mostly to computationally hard problems. Yet, exploiting the toolbox of parameterized complexity, we show several natural special cases admitting efficient algorithms. Our experiments suggest that many of these algorithms are applicable in practice. Overall, we contribute to a better understanding of fair allocation and multiwinner voting from both theoretical and practical perspectives.

ISBN 978-3-7983-3215-7 (print)
ISBN 978-3-7983-3216-4 (online)

ISBN 978-3-7983-3215-7

https://verlag.tu-berlin.de