TU WIEN Informatics

Classical Problems, New Models

Cooperation and Reconfiguration in Collective Decision-Making

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

DI Christian Hatschka, BSc
Matrikelnummer 01525634

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dr. Jiehua Chen

Diese Dissertation haben begutachtet:

Péter Biró	Piotr Skowron

Wien, 8. Oktober 2026

Christian Hatschka

TU WIEN Informatics

Classical Problems, New Models

Cooperation and Reconfiguration in Collective Decision-Making

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

DI Christian Hatschka, BSc
Registration Number 01525634

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dr. Jiehua Chen

The dissertation has been reviewed by:

Péter Biró	Piotr Skowron

Vienna, October 8, 2026

Christian Hatschka

Erklärung zur Verfassung der Arbeit

DI Christian Hatschka, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 8. Oktober 2026

Christian Hatschka

Danksagung

Anfangs möchte ich mich bei meiner Betreuerin, Associate Prof. Dr. Jiehua Chen, für ihre Unterstützung und ihren Rat ausdrücklich bedanken. Des Weiteren danke ich ihr dafür, dass sie mir die Teilnahme an Konferenzen, Workshops und Summer Schools ermöglicht hat, was meinen Horizont erweitert hat. Vor allem aber möchte ich ihr für ihre Geduld und ihre durchgehende Unterstützung während meines gesamten akademischen Werdegangs danken, die meine Erwartungen bei Weitem übertroffen hat.

Weiters möchte ich Dr. Sofia Simola dafür danken, dass sie mich zu Beginn meines Doktorats begleitet hat. Ohne ihre Ratschläge und die angenehmen Gespräche, die wir geführt haben, wäre mein Weg durch das Doktorat deutlich schwieriger gewesen.

Ebenso danke ich allen Koautorinnen und Koautoren meiner Publikationen für die angenehme Zusammenarbeit und die aufschlussreichen Diskussionen. Besonders hervorheben möchte ich Dr. Manuel Sorge und Dr. Martin Durand, deren Fachwissen mir geholfen hat, als Forscher zu wachsen.

Darüber hinaus danke ich meiner Freundin, meinen Eltern, meinen Großeltern und meinen Freunden für ihre Liebe und Unterstützung. Insbesondere danke ich meiner Freundin und meinem Vater, die meine Arbeit mehrfach Korrektur gelesen und mir geholfen haben, meinen Schreibstil zu verbessern.

Zu guter Letzt danke ich meinen Gutachtern dafür, dass sie sich die Zeit genommen haben, meine Dissertation zu lesen, und für ihr freundliches und wertvolles Feedback.

Acknowledgements

Firstly, I would like to thank my advisor, Associate Prof. Dr. Jiehua Chen, for her guidance and overall support. Furthermore, I would like to thank her for allowing me to attend conferences, workshops, and summer schools, which broadened my horizons. Most importantly, I would like to thank her for her patience and unwavering support throughout my academic journey, which went above and beyond my expectations.

Secondly, I would like to thank Dr. Sofia Simola for guiding me during the beginning of my PhD. Without her advice and the enjoyable conversations we shared, my PhD journey would have been much harder.

Thirdly, I would like to thank all coauthors of my papers for the pleasant cooperation and insightful discussions. I particularly want to highlight Dr. Manuel Sorge and Dr. Martin Durand, whose expertise helped me grow as a researcher.

Fourthly, I would like to thank my girlfriend, my parents, my grandparents, and my friends for their love and support. In particular, I would like to thank my girlfriend and father, who proofread my work multiple times and helped me improve my writing.

Lastly, I would like to thank my reviewers for taking time out of their schedules to read my thesis and for giving me kind and valuable feedback.

Kurzfassung

In dieser Dissertation schlagen wir neue Modelle für klassische Probleme der kollektiven Entscheidungsfindung vor, die im Kontext von Kooperation und Rekonfiguration auftreten. Wir untersuchen sie anhand etablierter Lösungskonzepte und analysieren die Berechnungskomplexität und die parametrisierte Komplexität von Problemen, die mit diesen Konzepten zusammenhängen.

Wir betrachten Kooperation mit Hilfe kooperativer Spiele: Agenten schließen bindende Verträge darüber ab, wie durch Zusammenarbeit erzeugter Nutzen (oder Kosten) aufgeteilt werden soll. Ziel ist es, einen Vertrag zu finden, der stabil ist, in dem Sinne, dass kein einzelner Agent und keine Teilmenge von Agenten einen Anreiz hat, von ihm abzuweichen. Um solche Verträge zu finden, konzentrieren wir uns auf die etablierten Lösungskonzepte der individuellen Rationalität und der Kernstabilität. Individuelle Rationalität garantiert, dass kein Agent durch die Zusammenarbeit schlechter gestellt ist als durch alleiniges Handeln; Kernstabilität garantiert zusätzlich, dass keine Koalition gemeinsam abweichen kann, um den Nutzen jedes ihrer Mitglieder strikt zu verbessern. Bei der Rekonfiguration besteht das Ziel darin, für zwei gegebene Lösungen eines kombinatorischen Problems einen Rekonfigurationspfad zu finden: eine Folge von Zwischenlösungen, von denen jede der vorherigen nahe ist und die die gegebenen Lösungen verbindet.

In den Kapiteln 3 bis 5 behandeln wir Kooperation, und in Kapitel 6 Rekonfiguration.

Kapitel 3 führt partitionierte kombinatorische Optimierungsspiele (PCOGs) ein. Bei PCOGs besteht die Eingabe aus einer Menge von Agenten und einer kombinatorischen Struktur (typischerweise einem Graphen) mit einem fixierten Optimierungsziel auf dieser Struktur (z. B. der Bestimmung einer minimalen dominierenden Menge), wobei die Struktur unter den Agenten aufgeteilt ist. Der Nutzen einer beliebigen Koalition von Agenten wird durch die optimale Lösung für den Teil der Struktur bestimmt, den sie besitzt. Wir untersuchen die Komplexität der Frage, ob ein kernstabiler Nutzenvektor—eine Aufteilung des Nutzens unter den Agenten, sodass keine Teilmenge von Agenten einen Anreiz zum Abweichen hat—existiert, sowie der Frage, ob ein gegebener kernstabil ist, für die PCOGs, die sich aus vier klassischen Graphproblemen ergeben: Minimum Vertex Cover, Minimum Dominating Set, Minimum Cost Spanning Tree und Maximum Matching. Wir liefern eine vollständige Komplexitätsklassifikation dieser Probleme, einschließlich einer parametrisierten Analyse hinsichtlich der Anzahl der Agenten und der maximalen Agentengröße.

In Kapitel 4 führen wir individuelle Rationalität in *multi-organizational scheduling* ein, bei der jede Organisation eine Menge identischer Maschinen und eine Menge von Aufträgen besitzt. Ein Schedule ist eine Zuordnung von Aufträgen zu Zeitfenstern auf den Maschinen. Er ist individuell rational, wenn keine Organisation schlechter gestellt ist, als wenn sie nur ihre eigenen Maschinen verwenden würde. Wir untersuchen die Komplexität der Frage, ob ein individuell rationaler Schedule existiert, der ein globales Optimierungsziel erreicht. Wir verwenden zwei Maße aus der Literatur: die Fertigstellungszeit des letzten Auftrags (*Makespan* $\mathsf{C}_{\mathsf{max}}$) und die Summe der Fertigstellungszeiten (ΣC). Wir präsentieren einen Überblick über die Berechnungskomplexität und die parametrisierte Komplexität für beide Maße. Wir untersuchen sowohl im Scheduling übliche Parameter als auch solche, die spezifisch für unsere Probleme sind.

In Kapitel 5 führen wir einen kooperativ-spieltheoretischen Rahmen für zustimmungsbasierte Wahlen mit mehreren Gewinnern (*multi-winner approval voting*) ein. Bei *multi-winner approval voting* ist eine Menge von Wählern mit Zustimmungen über einer Menge von Alternativen gegeben, und das Ziel besteht darin, ein Komitee—eine Teilmenge der Alternativen—fester Größe auszuwählen. Jede Wahlregel induziert ein Spiel, in dem die Wähler die Spieler sind: Eine Koalition von Wählern kann ein Komitee wählen, dessen Größe proportional zur Größe der Koalition ist, und jedes solche zulässige Komitee liefert einen Nutzenvektor für die Koalition. Wir untersuchen zwei Modelle: eines mit übertragbarem Nutzen, bei dem die Wähler den von einem Komitee erzielten Gesamtnutzen frei aufteilen dürfen, und eines mit nicht übertragbarem Nutzen, bei dem die Nutzen der Wähler direkt durch ein zulässiges Komitee realisiert werden müssen. Wir identifizieren eine hinreichende Bedingung für die Existenz kernstabiler Nutzenvektoren im Modell mit übertragbarem Nutzen. Für das Modell mit nicht übertragbarem Nutzen zeigen wir eine interessante Korrespondenz zwischen dem etablierten Konzept *justified representation* und den kernstabilen Elementen für die Wahlregel CC. Außerdem untersuchen wir die Berechnungskomplexität der Frage, ob ein kernstabiler Nutzenvektor existiert, sowie der Frage, ob ein gegebener kernstabil ist, für vier prominente Wahlregeln: AV, SAV, CC und PAV.

In Kapitel 6 untersuchen wir Rekonfiguration bei *multi-winner approval voting*: Für zwei gegebene Komitees der Größe k fragen wir, ob ein Rekonfigurationspfad aus Zwischenkomitees existiert, der sie verbindet und bei dem sich je zwei aufeinanderfolgende Komitees auf dem Pfad nur in wenigen Alternativen unterscheiden und jedes Zwischenkomitee unter der gewählten Wahlregel eine Mindestqualitätsschwelle erreicht. Wir zeigen, dass für alle Wahlregeln, die den Score eines Komitees als Summe der Scores einzelner Alternativen berechnen, ein kürzester Rekonfigurationspfad in Polynomialzeit mit einem einfachen Greedy-Algorithmus gefunden werden kann. Wir liefern eine feingranulare Komplexitätsanalyse des Rekonfigurationsproblems, das durch die vier im vorigen Kapitel untersuchten Wahlregeln induziert wird.

Abstract

In this thesis, we propose new models for classical problems in collective decision-making, arising in the context of cooperation and reconfiguration. We study them under established solution concepts and analyze the computational and parameterized complexity of computational problems related to these concepts.

We approach cooperation through cooperative games: Agents make binding contracts about how to share the utility (or cost) generated by working together. The goal is to find a contract that is stable, in the sense that no agent or subset of agents has an incentive to deviate from it. To this end, we focus on the established solution concepts of individual rationality and core stability. Individual rationality guarantees that no agent is worse off by collaborating than by acting alone; core stability additionally guarantees that no coalition can collectively deviate to strictly improve the utility of each of its members. In reconfiguration, given two solutions to a combinatorial problem, the goal is to find a reconfiguration path: a sequence of intermediate solutions, each close to the previous one, that connects the given solutions.

Chapters 3 to 5 deal with cooperation, and Chapter 6 deals with reconfiguration.

Chapter 3 introduces partitioned combinatorial optimization games (PCOGs). In PCOGs, the input consists of a set of agents and a combinatorial structure (typically a graph) with a fixed optimization goal on this structure (e.g., finding a minimum dominating set) such that the structure is divided among the agents. The utility of any coalition of agents is determined by the optimal solution for the part of the structure it owns. We study the complexity of deciding whether a core-stable utility vector—a division of utility among the agents such that no subset of agents has an incentive to deviate—exists, and of verifying whether a given one is core-stable, for the PCOGs derived from four classical graph problems: Minimum Vertex Cover, Minimum Dominating Set, Minimum Cost Spanning Tree, and Maximum Matching. We provide a complete complexity classification of these problems, including a parameterized analysis with respect to the number of agents and the maximum agent size.

In Chapter 4, we introduce individual rationality into multi-organizational scheduling, where each organization owns a set of identical machines and a set of jobs. A schedule is an assignment of jobs to time slots on the machines. It is individually rational if no organization is worse off than if it were to use only its own machines. We investigate the

complexity of determining whether there exists an individually rational schedule that achieves a global optimization goal. We use two measures from scheduling: the makespan (the completion time of the last job $\mathsf{C}_{\mathsf{max}}$) and the sum of completion times (ΣC). We give a computational and parameterized complexity overview for both measures. We study parameters common in scheduling as well as ones specific to the multi-organizational setting.

In Chapter 5, we introduce a cooperative game-theoretic framework for multi-winner approval voting. In multi-winner approval voting, we are given a set of voters with approval preferences over a set of alternatives, and the goal is to select a committee—a subset of alternatives—of fixed size. Each voting rule induces a game, where the voters are players: A coalition of voters can choose a committee whose size is proportional to the coalition's size, and each such admissible committee yields a utility vector for the coalition. We study two models: one with transferable utility, where voters may freely divide the total utility gained by a committee, and one with non-transferable utility, where voters' utilities must be directly realized by an admissible committee. We identify a sufficient condition for the existence of core-stable utility vectors under the transferable-utility model. For the non-transferable utility model, we show an interesting correspondence between the established notion of *justified representation* and elements in the core for the voting rule CC. We also investigate the computational complexity of deciding whether a core-stable utility vector exists, and of verifying whether a given one is core-stable, for four prominent voting rules: AV, SAV, CC, and PAV.

Finally, in Chapter 6, we study reconfiguration in multi-winner approval voting: Given two size-k committees, we ask whether there exists a reconfiguration path of intermediate committees connecting them, where each consecutive pair on the path differs by only a few alternatives and each intermediate committee meets a minimum quality threshold under the chosen voting rule. We show that for all voting rules that sum up the scores of individual alternatives to compute the score of a committee, a shortest reconfiguration path can be found in polynomial time via a simple greedy approach. We provide a fine-grained complexity analysis of the reconfiguration problem induced by the four voting rules studied in the previous chapter.

Contents

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Introduction **1**
1.1 Thesis Structure 4
1.2 Publications 6

2 Preliminaries **9**
2.1 Basic Notation and Graph-Theoretical Concepts 9
2.2 Computational Complexity 12
2.3 Algorithmic Techniques 20
2.4 Cooperative Games 26
2.5 Scheduling 31
2.6 Multi-Winner Approval Voting 32
2.7 Problem Compendium Organized by Complexity 34

3 Partitioned Combinatorial Optimization Games **41**
3.1 Introduction 41
3.2 Preliminaries 46
3.3 Partitioned Minimum Vertex Cover Game 54
3.4 Partitioned Minimum Dominating Set Game 79
3.5 Partitioned Maximum Matching Game and Partitioned Minimum Spanning Tree Game 94
3.6 Conclusion 97

4 Multi-Organizational Scheduling **99**
4.1 Introduction 99
4.2 Preliminaries 103
4.3 Minimizing the Makespan 105
4.4 Minimizing the Sum of Completion Times 136
4.5 Conclusion 143

5 Multi-Winner Games **145**
5.1 Introduction . . . 145
5.2 Preliminaries . . . 152
5.3 Non-Emptiness and Structure of the Core . . . 156
5.4 Computational Issues . . . 170
5.5 Conclusion . . . 196

6 Multi-Winner Reconfiguration **199**
6.1 Introduction . . . 199
6.2 Preliminaries . . . 204
6.3 Complexity Results . . . 206
6.4 Conclusion . . . 227

7 Conclusion **229**

Overview of Generative AI Tools Used **233**

Übersicht verwendeter Hilfsmittel **235**

List of Figures **237**

List of Tables **239**

Glossary **241**

Bibliography **245**

CHAPTER 1

Introduction

Collective decision-making aggregates the preferences or interests of individual agents into a joint outcome. We study two challenges about such outcomes:

1. How can agents reach a joint outcome under which cooperation is self-enforcing—one that no individual or coalition has reason to reject?
2. How can an existing outcome be reconfigured to a new one dynamically, through a sequence of small, controlled changes?

For both challenges, we ask not only whether such outcomes exist, but also how hard they are to find and to verify. To make these challenges concrete, we consider four scenarios that members of a university may need to navigate. The first three scenarios correspond to the first challenge: A set of agents seeks to make a collective decision that encourages cooperation. We approach this challenge through the framework of *cooperative games*. We use two solution concepts to identify the outcomes we seek. *Individual rationality* requires that no agent can improve its utility by working on its own. *Core stability* is even stronger, in that it requires that no subset of agents can collectively deviate to strictly improve the utility of every one of its members.

Consider first the institutes at a Viennese university, whose buildings are spread across the city. Each institute owns several buildings, and due to the rising number of students, the university requires that each institute building be close to at least one large lecture hall. An institute could meet this requirement by building lecture halls so that one is reachable from each of its buildings. However, a lecture hall placed close to the buildings of several institutes can be shared among them. By coordinating where to build the lecture halls, the institutes can build fewer of them in total and save space and money. While the benefit of cooperation is clear, it is unclear how the institutes should divide the costs among themselves. If each institute pays for the lecture halls that are built on

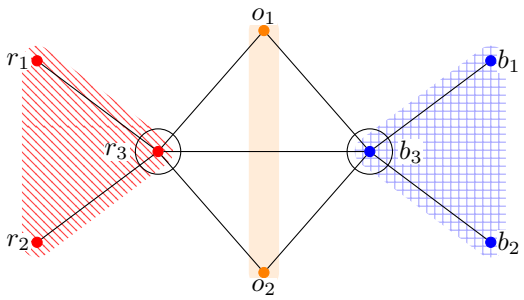


Figure 1.1: An instance of the lecture-hall scenario, drawn as a graph. Each vertex represents an institute building, and an edge connects two buildings that are close enough to share a lecture hall. The three institutes are distinguished by color, and the differently shaded region of each color encloses the buildings owned by the corresponding institute. The two circled buildings r_3 and b_3 form a minimum dominating set: placing a lecture hall at r_3 and at b_3 puts every building within reach of a lecture hall. Both halls serve buildings of all three institutes, so two shared halls suffice, whereas four halls would be needed if each institute covered its own buildings alone.

its own grounds, an institute whose lecture hall is used by a neighbor ends up subsidizing that neighbor. This raises the following question: Can the institutes agree on a split of the total cost such that no group of institutes has a reason to reject it? In this scenario, the institutes are the agents, a way of splitting the cost is the joint outcome, individual rationality guarantees that no institute pays more than it would alone, and core stability guarantees that no group of institutes can deviate and find a better outcome by only working among themselves. We model the first scenario in Chapter 3 as a partitioned dominating set game. Each building is a vertex of a graph, each institute owns the vertices that correspond to its buildings, and a dominating set corresponds to a choice of lecture-hall locations that covers every building; see Figure 1.1 for an example.

The same institutes face a second decision. Each of them owns a large cluster of machines that its members use to run simulations and other computationally intensive jobs. To deal with the increasing complexity of these jobs, the institutes want to buy additional machines for their clusters. Due to federal budget cuts, the university does not have the funds available, so it suggests an alternative approach: By connecting their clusters into a shared pool, the institutes can balance their load and complete their jobs sooner. As a result, the institutes would have access to more computing power without needing to buy additional machines for the clusters. While this seems like a sensible approach, an institute will only connect its cluster to the shared pool if doing so does not hurt it. More precisely, an institute will join the pool only if its jobs finish no later than they would if the institute had not joined the pool. If its jobs finish later, the institute has every reason to withdraw: It would be providing computing power to other institutes without receiving any benefit. Of course, if every institute simply runs its jobs on its own machines after joining the pool, then no institute has a reason to leave, but then there would be no reason for the pool to exist. This raises the following question: How

can we schedule the jobs among all the pooled clusters such that no institute is worse off than it would be alone, while also meeting a target on the overall completion time? In this scenario, the institutes are the agents, a way to schedule the jobs to the machines is an outcome, and individual rationality guarantees that no institute's jobs finish later in the shared pool than they would on its own cluster. We model the second scenario in Chapter 4 by introducing individual rationality into the multi-organizational scheduling framework. In this framework, each organization has a set of machines and jobs, and the organizations aim to share their machines to schedule their jobs more efficiently. In our scenario, the institutes are the organizations, and the machines in their clusters are the ones that are pooled.

However, the staff is not the only group at the university faced with collective decisions. Every year, the university gives teaching awards to a fixed number of courses, with the winners determined by the students' votes. The resulting winners should represent the student body fairly. Of course, not everyone is going to fully agree with the winning courses, but if a sufficiently large group of students finds that none of the courses they approved received an award, that group may complain that the awards ignored them, undermining the awards' purpose. For example, if there are four awards, a quarter of the students should be able to choose one award winner. If they all approve a common course, and none of the courses they approve wins, they are justified in complaining. This raises the following question: Can we select winners such that no sufficiently large group of students has cause to complain? In this scenario, the voters are the agents, a set of winning courses is a joint outcome, and core stability guarantees that no group of students is left underrepresented enough to have cause to complain. We model the third scenario in Chapter 5 by introducing multi-winner games. Here, the voters gain utility depending on which alternatives are selected as winners, and a group of voters can deviate if there is an alternative committee—a subset of alternatives of a size proportional to the group's size—that every member of the group strictly prefers to the current one. In our example, the students are the voters, the courses are the alternatives, and sets of courses are committees.

The fourth scenario corresponds to the second challenge: An outcome needs to be reconfigured to a new one, using small, controlled changes. We approach this challenge using the framework of *reconfiguration*, in which we are given an initial solution and a target solution to a combinatorial problem and ask for a reconfiguration path—a sequence of intermediate solutions, each differing only slightly from the previous one—that connects them.

The university is governed by an elected body of representatives. Due to term limits, the governing body should be replaced over time. If the university were to replace all representatives at once, it would lose substantial expertise instantly. To prevent this, the university decides to exchange only a few representatives at a time. At the same time, these intermediate bodies should also have the support of the university staff, i.e., they should represent the preferences of the voters. This raises the following question: Given the body's current composition and the desired future composition, is it possible to move

from the former to the latter, while replacing only a few representatives in each step, and while keeping the support of the staff? In this scenario, the combinatorial problem is a voting problem; elected bodies that have enough support from the staff are the solutions, and the current and desired composition of the body are the given initial and target solutions. The requirement that each step replaces only a few representatives ensures that consecutive solutions only differ slightly. We model the fourth scenario in Chapter 6 by introducing multi-winner reconfiguration. Here, we are given two committees—i.e., subsets of alternatives—of fixed size, and aim to connect them using a reconfiguration path consisting of committees that satisfy a quality threshold, where adjacent committees differ in at most a fixed number of alternatives. In our example, each possible composition of the governing body is a committee, the representatives are the alternatives, and the staff's support is the quality threshold.

Across this thesis, we study three classical settings—combinatorial optimization games, scheduling, and voting—from three different research areas: cooperative game theory, operations research, and computational social choice. We investigate the existence of solutions in our frameworks and analyze the computational complexity of problems related to our solution concepts.

Due to the demanding nature of our solution concepts, the associated computational problems are rarely tractable. Most of the problems we study are computationally hard, and hence admit no polynomial-time algorithm under current complexity assumptions. We conduct a fine-grained complexity analysis to identify islands of tractability. To this end, we use *parameterized complexity*, which classifies algorithmic difficulty in terms of problem parameters such as the solution size or the number of agents.

1.1 Thesis Structure

In Chapter 2, we introduce fundamental concepts and notation used throughout this thesis. We cover basic graph theory, computational complexity, the algorithmic techniques of integer linear programming and the ellipsoid method, cooperative games, and multi-winner approval voting. We provide a compendium of problems used in reductions. Each subsequent chapter additionally introduces concepts specific to its own setting.

Chapters 3 to 5 deal with the challenge of making a collective decision that encourages cooperation, and Chapter 6 deals with the challenge of reconfiguring an existing solution to a new one dynamically.

In Chapter 3, we introduce *partitioned combinatorial optimization games (PCOGs)*, in which a combinatorial structure (typically a graph) with a fixed optimization goal on this structure is partitioned among a set of agents, and the value available to a coalition of agents is the value of the optimal solution restricted to the part of the structure the coalition owns. An example of such a fixed optimization goal is the Dominating Set problem, as described in the first scenario with the lecture halls. PCOGs are a generalization of combinatorial optimization games, in which each agent

owns exactly one element of the underlying structure. We study Core-Stability Existence (CE), which asks whether a core-stable utility vector exists, and Core-Stability Verification (CV), which asks whether a given utility vector is core-stable. We investigate PCOGs derived from four classical graph problems: Minimum Vertex Cover (MinVC), Minimum Dominating Set (MinDS), Minimum Cost Spanning Tree (MinST), and Maximum Matching (MaxMatch). We also observe that an individually rational solution always exists for the problems we study. We make several structural observations about PCOGs and give a complete complexity overview for the studied problems. For the two intractable problems MinVC and MinDS, we show that CV is DP-complete and CE is Θ_2^{P}-complete. The two tractable problems behave differently: For MinST a core-stable utility vector always exists and can be computed in polynomial time, whereas for MaxMatch we complement the known coNP-hardness of CV with a matching coNP-containment. We also study two structural parameters, the number of agents and the maximum agent size, and give a complete parameterized complexity overview.

In Chapter 4, we introduce individual rationality into *multi-organizational scheduling*, in which each organization owns a set of identical machines and a set of jobs. A schedule assigns all jobs across the organizations' machines, and is *individually rational* if no organization is worse off under the joint schedule than under the schedule it could achieve using only its own machines. Such a schedule always exists—every organization can fall back on its local schedule—so the real challenge is to find one that additionally meets a global bound on the objective. We consider two objectives: the makespan (the time at which the last job completes) and the sum of completion times. We study the computational and parameterized complexity of this problem for both objectives. The two objectives sit at different levels in the complexity hierarchy. The decision variant we study is Θ_2^{P}-complete for the makespan but only NP-complete for the sum of completion times. This mirrors the classical single-organization setting, where minimizing the makespan is already NP-hard, while the sum of completion times can be minimized in polynomial time. To complement the intractability results, we provide multiple parameterized algorithms. One of our main results is a fixed-parameter tractable algorithm for the makespan with respect to the number of organizations plus the maximum processing time. We are also able to utilize a similar approach to present an XP algorithm with respect to the maximum processing time.

In Chapter 5, we introduce *multi-winner games*, a cooperative game-theoretic framework for multi-winner voting. We focus on the restricted case of multi-winner approval voting. In multi-winner approval voting, we are given a set of voters, each of whom has approval (binary) preferences over a set of alternatives, and the goal is to select a committee—a subset of alternatives—of fixed size. In multi-winner games, the players are the voters, each coalition is limited to a proportional seat cap and may only propose admissible committees up to that size, and each admissible committee induces a utility vector over the coalition members under a fixed voting rule. We study the games that arise from four prominent multi-winner approval voting rules: Approval Voting (AV),

Satisfaction Approval Voting (SAV), Chamberlin–Courant (CC), and Proportional Approval Voting (PAV). We study both the transferable-utility (TU) model, in which voters may freely divide the total utility induced by a committee, and the non-transferable-utility (NTU) model, in which voters' utilities must be directly realized by an admissible committee. Under TU, we show that the core is always non-empty and a core utility vector can be computed in polynomial time for AV and SAV—in fact for every totally separable rule, in which each alternative contributes independently to a voter's score—and we give a second, non-constructive non-emptiness proof via the Bondareva–Shapley theorem. This contrasts with the fact that for CC and PAV, the TU-core can be empty. Under NTU, we show that the core of CC coincides with the set of utility vectors realized by committees satisfying the fairness concept of justified representation. This directly implies that a core utility vector can be found in polynomial time for CC under NTU. We additionally show an equivalence between the NTU-core elements under AV, SAV, and PAV. We also provide a complexity overview of Core-Stability Existence and Core-Stability Verification for both models and all four rules.

In Chapter 6, we apply the reconfiguration framework to multi-winner approval voting. Given two size-k committees, we ask whether there exists a reconfiguration path of intermediate size-k committees, each consecutive pair differing by only a few alternatives, with every intermediate committee meeting a minimum quality threshold under the chosen voting rule. We again consider the prominent voting rules AV, SAV, CC, and PAV. We investigate the computational and parameterized complexity of determining the existence of a reconfiguration path and provide algorithms for finding a shortest reconfiguration path when one exists. For AV and SAV—and, more generally, for any rule whose committee score is the sum of the scores of its individual alternatives—we provide a simple greedy strategy that not only decides whether a reconfiguration path exists but also returns a shortest one in polynomial time. For CC and PAV, we show that deciding whether a reconfiguration path exists is PSPACE-complete. This mirrors the classical voting setting, for which computing a winning committee is tractable for AV and SAV, whereas finding a winning committee for CC and PAV is NP-hard. For CC and PAV, we investigate the parameterized complexity and obtain fixed-parameter tractability with respect to the number of alternatives, and for CC even with respect to the number of voters. For PAV, the number of voters also turns out to be a powerful parameter: Combined with any other of our studied parameters, it yields fixed-parameter tractability.

1.2 Publications

Below, I list the publications I produced during my time at TU Wien, along with a description of my individual contributions. I start by listing the publications that are a part of this thesis before moving on to those not included in it.

1.2.1 Publications in the Thesis

The following publications are listed in order of appearance.

Chapter 3. This chapter is based on the following publication:

> [CHS25]: Jiehua Chen, Christian Hatschka, and Sofia Simola. Partitioned combinatorial optimization games. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*, volume 413, pages 1429–1436, 2025

This publication went through multiple iterations. Initially, my supervisor suggested studying individual rationality in a setting where the cost or utility of an agent depends on the number of vertices belonging to a solution. Through a literature review, we moved on to the setting of partitioned combinatorial optimization games. Together with my collaborators, I helped design the model and find computational complexity results. Most results were discovered in close collaboration, although I had a stronger focus on the game induced by the Dominating Set problem; see Section 3.4.

Chapter 4. This chapter is based on the following publication:

> [CDH25]: Jiehua Chen, Martin Durand, and Christian Hatschka. Multi-organizational scheduling: Individual rationality, optimality, and complexity. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI 2025)*, pages 3780–3787, 2025

My colleague Martin Durand had worked on multi-organizational scheduling before; however, his work had focused more on fairness under approximation. Since my supervisor and I had worked on individual rationality in another setting, we suggested introducing it to the scheduling setting. This led to the formulation of the problem. As a team, we developed the framework based on Martin's previous work. We discovered most results in close cooperation. I had particular impact on the fixed-parameter tractability (FPT) result with respect to the maximum processing time and the number of organizations, by proving the necessary lemmas and formulating the integer linear program (ILP).

Chapter 5. This chapter is based on a currently unpublished paper that I co-authored with my supervisor. Through conferences and literature review, I had extensive exposure to the notion of proportionality in multi-winner voting. In particular, the notion of the NTU-core piqued my interest. As the core is a central concept in cooperative game theory, I proposed to study multi-winner voting from a game-theoretic perspective. The model and all results were the result of extensive discussions between my supervisor and me. In particular, I provided the basis for the proof utilizing the Bondareva–Shapley theorem and the polynomial-time algorithm for constructing a utility vector in the TU-core for totally separable voting rules.

Chapter 6. This chapter is based on the following publication:

[CHS24]: Jiehua Chen, Christian Hatschka, and Sofia Simola. Multi-winner reconfiguration. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems (NeurIPS 2024)*, pages 85899–85931, 2024

My supervisor suggested reconfiguration as a topic of interest. As I had previous experience with multi-winner voting, I suggested applying the reconfiguration framework to this setting. We developed the framework and found the results in close collaboration. In particular, I discovered the greedy approach to efficiently solving the reconfiguration problem for AV and SAV.

1.2.2 Other Publications

The following publications were omitted from my thesis.

[CHS23]: Jiehua Chen, Christian Hatschka, and Sofia Simola. Efficient algorithms for Monroe and CC rules in multi-winner elections with (nearly) structured preferences. In *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI 2023)*, volume 372, pages 397–404, 2023

[ACH+26]: Clemens Anzinger, Jiehua Chen, Christian Hatschka, Manuel Sorge, and Alexander Temper. How hard is it to explain preferences using few Boolean attributes? In *Proceedings of the 40th AAAI Conference on Artificial Intelligence (AAAI 2026)*, pages 16639–16646, 2026

[CHS26]: Jiehua Chen, Christian Hatschka, and Sofia Simola. Computational social choice: Parameterized complexity and challenges. *Computer Science Review*, 61:100835, 2026

CHAPTER 2

Preliminaries

This chapter introduces fundamental concepts and definitions used in this thesis. In Section 2.1, we explain some basic notation and introduce graph-theoretical concepts. Afterward, in Section 2.2, we give an overview of relevant complexity classes, as well as some necessary background knowledge for computational complexity. In Section 2.3, we present relevant algorithmic techniques that we use. In Section 2.4, we briefly introduce concepts from cooperative game theory. In Section 2.5, we give some necessary background on scheduling problems. In Section 2.6, we give an overview of multi-winner approval voting. Finally, in Section 2.7, we give a problem compendium sorted by complexity class. Concepts and definitions relevant to only a single chapter are introduced in that chapter.

2.1 Basic Notation and Graph-Theoretical Concepts

We use $\mathbb{N}$ to refer to the set of natural numbers, i.e., $\mathbb{N} = \{1, 2, 3, \ldots\}$. For the set of non-negative integers, we use $\mathbb{N}_0$. We use $[t]$ to denote the set consisting of all positive integers up to t, i.e., $[t] = \{x \in \mathbb{N} \mid x \leq t\}$. We use $\mathbb{R}$ for the set of real numbers. We use $\mathbb{R}_{\geq 0}$ to refer to the set of non-negative real numbers. Given a vector of length n, $\boldsymbol{x} = (x_1, \ldots, x_n)$, we use $\boldsymbol{x}_i$ to refer to the i-th entry of the vector. Given two sets X and Y, we use $X \Delta Y$ to denote the symmetric difference between X and Y, i.e., $X \Delta Y = (X \setminus Y) \cup (Y \setminus X)$.

Graphs are used throughout this thesis in multiple ways. In some cases, we study problems based on graphs, and in many other cases, we reduce from problems defined on graphs to show intractability. We focus on undirected graphs.

Definition 2.1. *An undirected graph (in short graph) is a pair (V, E), where V is a set of vertices and $E \subseteq \{\{v_i, v_j\} \mid v_i, v_j \in V : v_i \neq v_j\}$ is a set of edges.*

Remark 2.2. *Note that the condition $v_i \neq v_j$ holds because we only consider simple graphs, i.e., graphs that do not have multi-edges (multiple edges between the same pair of vertices) or loops (an edge from a vertex to itself).*

We usually use n and m to refer to the number of vertices and edges, respectively, i.e., $|V| = n$ and $|E| = m$. Throughout this chapter, we use G to refer to a graph in general, and V and E to refer to the set of vertices and edges of the graph G, respectively.

Definition 2.3 (Adjacency, neighborhood, degree)**.** *We call two vertices $v_i, v_j \in V$* adjacent *(or equivalently* neighbors*) to each other if there is an edge $\{v_i, v_j\} \in E$. We call a vertex $v_i \in V$* incident *to an edge $e \in E$ if $v_i \in e$; if a vertex $v_i \in V$ is incident to an edge $e \in E$, we also call v_i an* endpoint *of e. The* open neighborhood *$N_G(v)$ of a vertex $v \in V$ is the set of its neighbors, i.e., $N_G(v) = \{w \mid w \in V : \{v, w\} \in E\}$. The* closed neighborhood *$N_G[v]$ of a vertex $v \in V$ is the set containing its neighbors and itself, i.e., $N_G[v] = N_G(v) \cup \{v\}$. The* degree *$\deg(v)$ of a vertex v is the number of its neighbors, or in other words, the cardinality of its open neighborhood $|N_G(v)|$.*

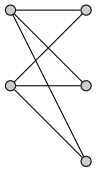
If the underlying graph G is clear from the context, we may omit G from the subscript, i.e., we write $N(v)$ and $N[v]$, respectively.

A *path* between two vertices $v_i, v_j \in V$ is a finite sequence of vertices $(v_{i_1}, \ldots, v_{i_k})$ such that $\{v_{i_\ell}, v_{i_{\ell+1}}\} \in E$ for all $\ell \in [k-1]$, all vertices are distinct, $v_{i_1} = v_i$, and $v_{i_k} = v_j$. We say that the edge $\{v_{i_\ell}, v_{i_{\ell+1}}\}$ is on the path for $\ell \in [k-1]$. The *length of a path* is the number of edges on the path, or equivalently, the number of vertices minus one. We say that a vertex $v_i \in V$ is *reachable* from a vertex $v_j \in V$ if there exists a path between v_i and v_j. A graph is *connected* if there exists a path between every two vertices $v_i, v_j \in V$.

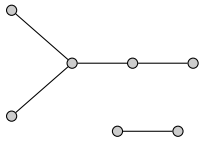
A *cycle* is a finite sequence of vertices $(v_{i_1}, \ldots, v_{i_k}, v_{i_1})$ with $k \geq 3$ such that v_{i_1} is the only repeating vertex and $\{v_{i_\ell}, v_{i_{\ell+1}}\} \in E$ for all $\ell \in [k]$, where $v_{i_{k+1}} = v_{i_1}$. We say that the edge $\{v_{i_\ell}, v_{i_{\ell+1}}\}$ is on the cycle for $\ell \in [k]$, where $v_{i_{k+1}} = v_{i_1}$. The *length of a cycle* is the number of edges on the cycle or, equivalently, the number of distinct vertices.

Definition 2.4 (Subgraphs)**.** *We say a graph $G_1 = (V_1, E_1)$ is a* subgraph *of another graph $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. For a subset $V' \subseteq V$ of vertices, the* subgraph (of G) induced by V' *is denoted by $G[V']$ and is defined as $G[V'] = (V', E')$, where $E' = \{\{v_i, v_j\} \mid v_i, v_j \in V', \{v_i, v_j\} \in E\}$. For a subset of edges $E' \subseteq E$, the* subgraph (of G) induced by E' *is denoted by $G[E']$ and is defined as $G[E'] = (V', E')$, where $V' = \bigcup_{e' \in E'} e'$.*

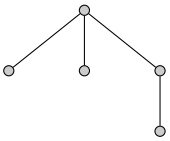
Informally, the subgraph induced by V' can be obtained by keeping only the vertices in V' and the edges between vertices in V', and the subgraph induced by E' can be obtained by keeping only the edges in E' and the vertices that are incident to an edge in E' from G.



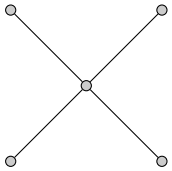
(a) Complete bipartite graph $K_{2,3}$.



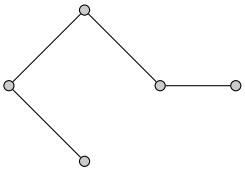
(b) Forest.



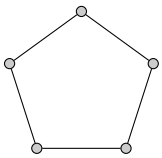
(c) Tree.



(d) Star.



(e) Path.



(f) Cycle.

Figure 2.1: Examples of the graph classes introduced in this subsection: (a) the complete bipartite graph $K_{2,3}$, (b) a forest, (c) a tree, (d) a star, (e) a path, and (f) a cycle. The circles are vertices, and the lines connecting the vertices are edges.

2.1.1 Relevant Graph Classes

We now discuss some commonly studied graph classes. A graph is *bipartite* if there exists a partition of the vertices V into two disjoint subsets V_1 and V_2 such that for all edges $\{v_i, v_j\} \in E$ it holds that $v_i \in V_1$ and $v_j \in V_2$, or $v_i \in V_2$ and $v_j \in V_1$. A graph is a *complete bipartite graph* if there exists a partition of the vertices V into two disjoint subsets V_1 and V_2 such that $E = \{\{v_i, v_j\} \mid v_i \in V_1, v_j \in V_2\}$. We call the complete bipartite graph with $|V_1| = i$ and $|V_2| = j$ $K_{i,j}$.

A graph is *acyclic* if there exists no cycle in the graph. A graph is a *forest* if it is acyclic. A graph is a *tree* if it is acyclic and connected. A graph is a *star graph* (or star for short) if there exists a vertex $v_i \in V$ such that $E = \{\{v_i, v_j\} \mid v_j \in V \setminus \{v_i\}\}$. A graph is a *path graph* (or *path* for short) if there exists an ordering of the vertices $v_1, \ldots, v_n$ such that $E = \{\{v_i, v_{i+1}\} \mid i \in [n-1]\}$. A graph is a *cycle graph* (or *cycle* for short) if there exists an ordering of the vertices $v_1, \ldots, v_n$ such that $E = \{\{v_i, v_{i+1}\} \mid i \in [n-1]\} \cup \{\{v_n, v_1\}\}$. See Figure 2.1 for examples of the aforementioned graph classes.

2.1.2 Graph Concepts

We now define six standard combinatorial objects on graphs: independent sets, dominating sets, vertex covers, cliques, matchings, and spanning trees.

Definition 2.5. *Let $G = (V, E)$ be an undirected graph. Let $V' \subseteq V$ be a subset of vertices and $E' \subseteq E$ be a subset of edges. We say that*

- *V' is an* independent set *of G if no two vertices in V' are adjacent.*

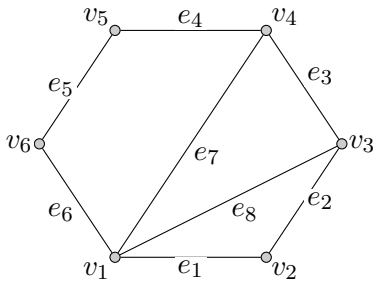


Figure 2.2: A graph $G = (V, E)$ with the 6 vertices $v_1, \ldots, v_6$ and the 8 edges $e_1, \ldots, e_8$.

- *V' is a dominating set of G if for every vertex $v \in V$ it holds that $N_G[v] \cap V' \neq \emptyset$. In other words, if v is not in V', then one of its neighbors is.*
- *V' is a vertex cover of G if every edge $e \in E$ has at least one endpoint in V', i.e., $e \cap V' \neq \emptyset$.*
- *V' is a clique of G if there is an edge between every pair of vertices in V', i.e., $\{\{v, w\} \mid v, w \in V' \colon v \neq w\} \subseteq E$.*
- *E' is a matching of G if no two distinct edges in E' share a common endpoint, i.e., for all $e_i, e_j \in E'$ such that $e_i \neq e_j$ it holds that $e_i \cap e_j = \emptyset$.*
- *a subgraph $G' = (V, E')$ of G is a spanning tree of G if G' is a tree, i.e., it is connected and acyclic. Note that the spanning tree contains all vertices in V by construction.*

Remark 2.6. *Note that if V' is an independent set, then $V \setminus V'$ is a vertex cover and vice versa. This further implies that the size of a maximum independent set and the size of a minimum vertex cover sum up to the number of vertices in the graph.*

Example 2.7. *Consider the graph depicted in Figure 2.2. A maximum independent set is given by $V_1' = \{v_2, v_4, v_6\}$, as there is no edge between any of these vertices and no larger independent set exists. $V_2' = \{v_1, v_5\}$ is a minimum dominating set, as v_1 dominates all vertices except v_5, and v_5 is a member of the dominating set. $V_3' = V \setminus V_1' = \{v_1, v_3, v_5\}$ is a minimum vertex cover by Remark 2.6. A maximum matching can have size at most 3, as there are 6 vertices and each edge covers exactly 2 vertices. A maximum clique is given by $V_4' = \{v_1, v_3, v_4\}$. An example of a maximum matching in the example graph is $E_1' = \{e_1, e_3, e_5\}$. By setting $E_2' = E \setminus \{e_6, e_7, e_8\}$, the graph $G' = (V, E_2')$ is a spanning tree.*

2.2 Computational Complexity

This section, until Section 2.2.3, follows the notation and definitions by Arora and Barak [AB09]. If we follow different definitions or notations, we note it at the point of use.

In Section 2.2.3, we follow the notation and definitions by Cygan et al. [CFK+15]. If we follow different definitions or notations, we note them at the point of use. For an alternative introduction to parameterized complexity, we refer to the book by Niedermeier [Nie06].

In this thesis, we focus on decision problems. These problems raise a question that can be answered with a yes or a no for a given instance. An example of a decision problem is the SAT problem.

Definition 2.8 (Boolean variables, Boolean formulas, CNF, SAT [AB09, Pap94])**.** *A* Boolean variable *is a variable that can either be True or False. We sometimes identify True as* 1 *and False as* 0*. A* Boolean formula *is a formula connecting Boolean variables using the logical connectives* $\neg$ negation, $\wedge$ conjunction, *and* $\vee$ disjunction*. Alternatively, a Boolean formula over Boolean variables* $\{x_1, \ldots, x_n\}$ *can be defined recursively: Every Boolean variable* x *is a Boolean formula; if* φ *and* ψ *are Boolean formulas, then so are* $\neg\varphi$*,* $(\varphi \wedge \psi)$*, and* $(\varphi \vee \psi)$*. A* truth assignment $\tau\colon \{x_1, \ldots, x_n\} \to \{True, False\}$ *is a function mapping Boolean variables to True and False. A truth assignment* τ satisfies *a Boolean formula* φ *if* φ *evaluates to True under* τ *by interpreting* $\neg, \wedge$*, and* $\vee$ *in the usual way. That is, a Boolean variable evaluates to True if it is mapped to True under* τ*,* $\neg\varphi$ *evaluates to True if* φ *evaluates to False,* $\varphi \wedge \psi$ *evaluates to True if both* φ *and* ψ *evaluate to True, and* $\varphi \vee \psi$ *evaluates to True if at least one of* φ *and* ψ *evaluates to True. We say a Boolean formula is* satisfiable *if there exists at least one truth assignment that satisfies it. If it is not satisfiable, we say that it is* unsatisfiable*. A* literal *is a Boolean variable* x_i *or its negation* $\neg x_i$*. A* clause *is a disjunction of literals, e.g.,* $x_1 \vee x_2 \vee \neg x_3$*. A Boolean formula is in* conjunctive normal form (CNF) *if it is a conjunction of clauses, e.g.,* $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3)$*. The SAT problem takes a CNF formula as input and asks whether it is satisfiable.*

Decision problems have a counterpart in languages. For a given finite set of symbols Σ, we use Σ^* to refer to the set of all finite strings over Σ. A *language* L is a subset of Σ^*. The *complement* of a language $L \subseteq \Sigma^*$ is $\overline{L} = \Sigma^* \setminus L$. By encoding the problem using the symbols in Σ, we can map all instances of a given decision problem to strings in Σ^*. The strings in Σ^* can then be partitioned into three disjoint subsets:

- Strings that encode a yes-instance of the decision problem.
- Strings that encode a no-instance of the decision problem.
- Strings that do not encode an instance of the decision problem.

The first subset of strings is the language L, which is associated with a decision problem Π. We assume that a fixed encoding scheme is given, and we use $L[\Pi]$ to refer to the language that is derived from the decision problem Π using this fixed encoding scheme. We assume a fixed binary encoding over $\Sigma = \{0, 1\}$ and the length $|I|$ of an instance is the number of bits in its encoding. A decision problem Π can be understood as asking if the encoding of an instance is part of the language $L[\Pi]$. Generally, we can determine in polynomial

time whether a string encodes an instance of the decision problem; hence, we assume every string is either a yes-instance or a no-instance. We identify a decision problem Π with its language $L[\Pi]$, and state complexity results about either interchangeably.

2.2.1 P, NP, and coNP

Informally, a complexity class is a way to group problems based on the amount of resources required to solve them [Joh90]. In this thesis, the complexity classes we consider group problems by the amount of time or space needed to solve them in the worst case. We first introduce the complexity classes P and NP.

Definition 2.9 (P)**.** *A decision problem Π is* polynomial-time solvable *and in the complexity class* P *if there exists a deterministic algorithm that, given an instance I of Π, can determine whether I is a member of the language $L[\Pi]$ in time that is polynomial in the length of I.*

We also say that problems in P are *tractable*.

Definition 2.10 (NP)**.** *A decision problem Π is in the complexity class* NP *if there exists a polynomial p such that an instance I is a yes-instance of Π if and only if there exists a* certificate *c of length at most $p(|I|)$ that can be checked in time polynomial in $|I|$. Equivalently, $\Pi \in$ NP if there exists a non-deterministic algorithm that determines whether I is a member of the language $L[\Pi]$ in time that is polynomial in the length of I.*

For the previously mentioned SAT problem, a non-deterministic algorithm would first guess a truth assignment and then verify whether it satisfies the given formula, and a certificate would be a satisfying assignment.

Definition 2.11 (Polynomial-time many-one reduction)**.** *Given two decision problems Π_1 and Π_2, a* polynomial-time many-one reduction *from Π_1 to Π_2 is an algorithm that, for every instance I of Π_1, returns an instance I' of Π_2 such that*

1. *I is a yes-instance of Π_1 if and only if I' is a yes-instance of Π_2, and*
2. *the algorithm runs in time polynomial in $|I|$.*

We say that a decision problem Π_1 can be reduced *to a decision problem Π_2 if there exists a polynomial-time many-one reduction from Π_1 to Π_2.*

We use many-one reductions throughout the chapters. Unless stated otherwise, *reduction* is used for polynomial-time many-one reductions.

We only consider complexity classes that contain P (such as NP, coNP, ...). For these complexity classes, hardness is defined via polynomial-time reductions. A problem Π is hard for one of our considered complexity classes if every problem in that complexity class can be reduced to Π.

Definition 2.12 (NP-hardness)**.** *A decision problem* Π *is* NP-hard *if every problem in* NP *can be reduced to it.*

Note that a problem can be NP-hard without being in NP. As concatenating two polynomial-time many-one reductions results in a polynomial-time many-one reduction, it suffices to provide a reduction from a single NP-hard problem to Π to show that Π is NP-hard.

Inside the class of NP, the hardest problems are referred to as NP-complete problems.

Definition 2.13 (NP-completeness)**.** *A problem is* NP-complete *if it is in* NP *and* NP*-hard.*

Hardness and completeness for the following complexity classes are defined analogously, using polynomial-time many-one reductions.

By definition, $\mathrm{P} \subseteq \mathrm{NP}$, but it remains an open question whether $\mathrm{P} = \mathrm{NP}$ or $\mathrm{P} \neq \mathrm{NP}$. However, it is generally assumed that $\mathrm{P} \neq \mathrm{NP}$.

Definition 2.14 (coNP)**.** *A decision problem* Π *is in the complexity class* coNP *if there exists a polynomial* p *such that an instance* I *is a no-instance of* Π *if and only if there exists a* certificate c *of length at most* $p(|I|)$ *that can be checked in time polynomial in* $|I|$*. Equivalently,* $\Pi \in \mathrm{coNP}$ *if there exists a non-deterministic algorithm that determines whether* I *is* not *a member of the language* $L[\Pi]$ *in time that is polynomial in the length of* I*.*

Equivalently, coNP is the class of all problems whose complements are in NP, i.e., a problem Π is in coNP if there exists a problem Π' in NP such that an instance I is a yes-instance of Π if and only if I is a no-instance of Π'. In terms of languages, it must hold that $L[\Pi] \cap L[\Pi'] = \emptyset$ and $L[\Pi] \cup L[\Pi']$ is precisely the set of encodings of instances in Π and Π'. The UNSAT problem asks whether a CNF formula has no satisfying assignment and is in coNP. It is the complementary problem to the aforementioned SAT problem.

By definition, $\mathrm{P} \subseteq \mathrm{coNP}$, but it remains an open question whether $\mathrm{P} = \mathrm{coNP}$ or $\mathrm{P} \neq \mathrm{coNP}$. However, it is generally assumed that $\mathrm{P} \neq \mathrm{coNP}$. Further, while it is unknown whether $\mathrm{NP} \neq \mathrm{coNP}$, it is generally assumed that $\mathrm{NP} \neq \mathrm{coNP}$.

We also consider classes of problems that are even harder to solve than problems in NP and coNP under current complexity assumptions.

2.2.2 Complexity Classes Beyond NP

We now define complexity classes that lie beyond NP and coNP. We start by defining an NP-oracle.

Definition 2.15 (NP-oracle)**.** *An* NP*-oracle is a black-box algorithm that, given an instance I of a problem Π in* NP*, returns, in a single time step, yes if I is a yes-instance, and no if I is a no-instance.*

Remark 2.16. *The* NP*-oracle can be instantiated for an arbitrary* NP*-complete problem (e.g., SAT), since every* NP *problem can be reduced to it in polynomial time.*

We can now introduce these classes in order of increasing complexity, starting with the complexity class DP (not to be confused with dynamic programming, which is also abbreviated as DP).

Definition 2.17 (DP [Pap94])**.** *A decision problem Π is in the complexity class* DP *if there exist two decision problems $\Pi_1 \in$ NP, $\Pi_2 \in$ coNP such that $L[\Pi] = L[\Pi_1] \cap L[\Pi_2]$.*

Note that a problem being in DP is not the same as being in the intersection of the classes coNP and NP. Indeed, DP contains both NP and coNP. To see how a DP problem decomposes, note that deciding DP-membership can be done by verifying that one NP-oracle query answers yes and another answers no. An example of a DP-complete problem is SAT-UNSAT, which takes two CNF formulas φ_1 and φ_2 as input and asks whether φ_1 is satisfiable and φ_2 is unsatisfiable. One can see that this can be solved by two NP-oracle calls, where the first checks the satisfiability of φ_1 and the second checks the unsatisfiability of φ_2.

Definition 2.18 (Θ_2^{P} [Wag90])**.** *A decision problem Π is in the complexity class Θ_2^P if it can be solved by a polynomial-time algorithm using logarithmically many* NP*-oracle calls.*

The complexity class Θ_2^{P} is also known as $\mathrm{P}^{\mathrm{NP[log]}}$ and $\mathrm{P}_{||}^{\mathrm{NP}}$. Initially, $\mathrm{P}_{||}^{\mathrm{NP}}$ was defined as the class of decision problems that can be solved with polynomially many parallel (non-adaptive) NP-oracle calls. It was shown that this coincides with the class of decision problems that can be solved using a logarithmic number of (adaptive) NP-oracle calls.

Theorem 2.19 ([Wag90])**.** $\Theta_2^P = \mathrm{P}^{\mathrm{NP[log]}} = \mathrm{P}_{||}^{\mathrm{NP}}$.

Many problems that ask whether an optimal solution with a given property exists fall into this class. An example of such a problem is Vertex Cover Membership, which asks whether there exists a minimum vertex cover that contains a given vertex. Informally, one can say that the hardness in this problem arises from a logarithmic number of NP-oracle calls being needed to determine the size of an optimal solution (using binary search) and another NP-oracle call to check whether an optimal solution with the property exists.

Definition 2.20 (Δ_2^{P} (also known as P^{NP}) [Pap94][1])**.** *A decision problem Π is in the complexity class Δ_2^P if it can be solved by a polynomial-time algorithm with access to an* NP*-oracle.*

[1] Papadimitriou referred to the complexity class as $\Delta_2\mathbf{P}$. We follow his definition, but use the now commonly used superscript notation Δ_2^{P}.

Note that this extends the class Θ_2^{P}, as Δ_2^{P} allows a polynomial number of NP-oracle calls. An example of a Δ_2^{P}-complete problem is determining if, in a lexicographically maximal satisfying truth assignment, the last variable is set to 1.[2] Before we introduce the next complexity class, we define *binary relations*:

Definition 2.21 (Binary relations)**.** *A binary relation R is a set of ordered pairs. A relation is* polynomial-time recognizable *if, given an ordered pair (x, y), it can be determined in polynomial time whether $(x, y) \in R$.*

This now allows us to define the next complexity class in two equivalent manners:

Definition 2.22 (Σ_2^{P} (also known as $\mathrm{NP}^{\mathrm{NP}}$))**.** *A decision problem Π is in the complexity class Σ_2^P if it can be solved by a non-deterministic polynomial-time algorithm with access to an* NP*-oracle. Equivalently, Π is in Σ_2^P if there exist a polynomial-time recognizable relation R and a polynomial p such that an instance I is a yes-instance of Π if and only if $\exists x \, \forall y \colon (I, x, y) \in R$, where x and y range over strings of length at most $p(|I|)$.*

An example of a Σ_2^{P}-complete problem is $\exists\forall$-SAT, which takes two disjoint sets X, Y of Boolean variables and a Boolean formula φ as input and asks if there exists a truth assignment over X such that φ is satisfied for every truth assignment of Y [Sto76].

Definition 2.23 (PSPACE, NPSPACE)**.** *A decision problem Π is in the complexity class* PSPACE*, if it can be solved by a deterministic algorithm using polynomial space in the length of the input. A decision problem Π is in the complexity class* NPSPACE*, if it can be solved by a non-deterministic algorithm using polynomial space in the length of the input.*

Savitch showed that every non-deterministic algorithm that uses polynomial space can be transformed into a deterministic algorithm that uses polynomial space. This gives us Savitch's Theorem:

Theorem 2.24 ([Sav70])**.** PSPACE = NPSPACE.

An example of a PSPACE-complete problem is Quantified Boolean Formula, which takes a *quantified Boolean formula*—a Boolean formula in which every variable is bound by an existential ($\exists$) or universal ($\forall$) quantifier—and asks whether the formula evaluates to True [SM73].

We note that the complexity classes satisfy the following inclusion schemes, with the inclusions being strict under current complexity assumptions:

[2] This problem was shown to be Δ_2^{P}-complete by Krentel [Kre88]. However, he did not name the problem in his work.

$$\mathrm{P} \subseteq \begin{array}{c}\mathrm{NP}\\ \diamond \\ \mathrm{coNP}\end{array} \subseteq \mathrm{DP} \subseteq \Theta_2^{\mathrm{P}} \subseteq \Delta_2^{\mathrm{P}} \subseteq \Sigma_2^{\mathrm{P}} \subseteq \mathrm{PSPACE} = \mathrm{NPSPACE}$$

2.2.3 Parameterized Complexity

We study multiple problems that are computationally hard. Parameterized complexity is a tool for further studying hard instances of problems and identifying the origins of their hardness. Sometimes, instances of a problem where a particular parameter is constant can be solved efficiently. For these parameters, it is also of interest how their growth affects the running time of an algorithm solving the problem.

Definition 2.25 (Parameterized problems)**.** *A* parameterized problem *is a language* $\Pi \subseteq \Sigma^* \times \mathbb{N}$ *for some fixed alphabet* Σ*. For an instance* (I, k)*, we refer to* k *as the* parameter*.*

For example, an instance of SAT parameterized by the number of variables is a pair (I, k), where I is some encoding of a Boolean formula and k is the number of variables in this formula. An instance then belongs to the yes-instances of this parameterized problem if it is satisfiable and has k variables.

Definition 2.26 (FPT)**.** *An* FPT-algorithm *(FPT stands for* Fixed-Parameter Tractable*) for a parameterized problem* Π *is an algorithm that solves every instance* (I, k) *of the problem in time* $f(k) \cdot |I|^{O(1)}$*, where* f *is a computable function and* $|I|$ *is the size of the instance. We also say that the algorithm runs in* FPT-time*. We say that the problem* Π *is in the complexity class* FPT *or that* Π *is FPT if it admits an FPT-algorithm.*

Vertex Cover[3], the problem of determining whether a vertex cover of size at most k exists for a given graph G, with the solution size k as a parameter, is FPT, since branching over uncovered edges iteratively solves the problem in time $2^k \cdot |I|^{O(1)}$.

Definition 2.27 (XP)**.** *An* XP-algorithm *(XP stands for slice-wise polynomial) for a parameterized problem* Π *is an algorithm that solves every instance* (I, k) *of the problem in time* $f(k) \cdot |I|^{g(k)}$*, where* f *and* g *are computable functions. We also say that the algorithm runs in* XP-time*. We say that the problem* Π *is in the complexity class* XP *or it is in XP if it admits an XP-algorithm.*

XP can be seen as the class of problems that are polynomial-time solvable for every constant parameter value. An example of a parameterized problem in XP is Independent Set, which asks whether there exists an independent set of size k, with the solution size

[3]See Section 2.7 for a formal definition of the problems mentioned.

k as the parameter. This follows, as we can branch over all size-k subsets of vertices in time $|V|^k$.

As every FPT-algorithm is an XP-algorithm where $g(k)$ is a fixed constant function, it follows that FPT $\subseteq$ XP. It is known that FPT $\subsetneq$ XP [DF13]. Note that this is an unconditional result, i.e., it does not rely on a complexity-theoretic assumption.

The *W-hierarchy* consists of the complexity classes FPT, W[1], W[2], ... The complexity classes *W[i]* for $i \in \mathbb{N}$ are defined through reducibility to special kinds of circuits, but we omit the exact definition as the circuit characterization is not needed in this thesis. The exact definition can be found in the book by Downey and Fellows [DF13]. The following inclusions are currently known:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{XP}$$

Analogous to polynomial-time reductions, we can use a parameterized reduction from a W[i]-hard problem to a parameterized problem Π to show that Π is W[i]-hard as well.

Definition 2.28 (Parameterized reduction [CFK+15])**.** *Let $\Pi_1, \Pi_2 \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A* parameterized reduction *from Π_1 to Π_2 is an algorithm that, given an instance (I, k) of Π_1, outputs an instance (I', k') of Π_2 such that*

1. *(I, k) is a yes-instance of Π_1 if and only if (I', k') is a yes-instance of Π_2,*
2. *$k' \leq g(k)$ for some computable function g, and*
3. *the running time of the algorithm is $f(k) \cdot |I|^{O(1)}$, where f is a computable function.*

A parameterized problem is *W[i]-hard* if every problem in W[i] parameterized-reduces to it, and *W[i]-complete* if it is additionally in W[i]. Although membership in the complexity class W[i] is not relevant in this thesis, a problem being W[i]-hard is relevant, as it is assumed that FPT $\neq$ W[1], and therefore a problem being W[i]-hard makes it unlikely that the problem is FPT. An example of a W[1]-complete parameterized problem is Independent Set, with the solution size k as the parameter. An example of a W[2]-complete parameterized problem is Dominating Set, with the solution size k as the parameter.

Note that while a W[i]-hard problem is unlikely to be FPT, it can very well still be in XP. As mentioned earlier, an XP-problem can be interpreted as the underlying problem being in P when the parameter is a constant. For many parameterized problems, however, the problem could remain NP-hard even for a constant parameter value.

Definition 2.29 (para-NP-hard [FG06])**.** *We call a parameterized problem* para-NP-hard*, if it remains* NP*-hard even if the parameter value is a fixed constant.*

Note, however, that the problem may be polynomial-time solvable for a certain constant parameter value and still be para-NP-hard, as long as there is some constant value the parameter can take for which the problem remains NP-hard.

2.3 Algorithmic Techniques

In this section, we give background information on some of the more intricate algorithmic techniques we use.

2.3.1 Integer Linear Programming and Linear Programming

Integer Linear Programming is a widely applied technique for modeling and solving optimization problems with linear constraints over integer-valued variables. This subsection follows the notation in the book by Conforti et al. [CZC14].

Definition 2.30 (Integer linear programs (ILPs) and solutions)**.** *An integer linear program*[4] *(ILP for short) is of the form:*

$$\max \boldsymbol{c} \cdot \boldsymbol{x} \tag{2.1}$$

$$\text{subject to } A\boldsymbol{x} \leq \boldsymbol{b} \tag{2.2}$$

$$\boldsymbol{x} \geq 0 \text{ integral} \tag{2.3}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)$ is an n-dimensional vector of integer variables to be determined, $\boldsymbol{c} = (c_1, \ldots, c_n)$ is a rational-valued vector, $A = (a_{ij})$ is a rational-valued $m \times n$ matrix, and $\boldsymbol{b} = (b_1, \ldots, b_m)$ is a rational-valued vector. An n-dimensional vector $\boldsymbol{x}$ is integral if $\boldsymbol{x} \in \mathbb{Z}^n$. A feasible solution to an ILP is an integral vector $\boldsymbol{x}$ that satisfies (2.2) and (2.3). An optimal solution to an ILP is a feasible solution that maximizes the objective function.

Remark 2.31. *The vectors $\boldsymbol{b}$ and $\boldsymbol{x}$ are considered as column vectors and $\boldsymbol{c}$ as a row vector.*

The *ILP Feasibility* problem asks whether a given ILP has a feasible solution. For instances of the ILP Feasibility problem, the objective function to be maximized is usually omitted. Throughout this thesis, we are mainly interested in the ILP Feasibility problem.

Note that Integer Linear Programming can also solve minimization problems, as well as allow for equality ($=$) and greater than or equal ($\geq$) in its constraints. Solving ILP Feasibility is NP-hard [Pap81]; however, ILP solvers such as Gurobi [Gur26] and CPLEX [IBM26] are capable of solving instances efficiently in practice.

ILP is usually used in this thesis as a tool to show that a parameterized problem is FPT, as Lenstra showed that every ILP Feasibility instance can be solved in FPT-time with respect to the number of (integer) variables [Len83]. The running time of the procedure was improved upon by Kannan [Kan87], and Frank and Tardos [FT87], leading to the following theorem:

[4] Conforti et al. call them pure integer linear programs to distinguish them from mixed integer linear programs (MILP), which also allow non-integer variables.

Theorem 2.32 ([Len83, Kan87, FT87]). *An ILP* Feasibility *instance can be solved in time $O(n^{2.5n+o(n)} \cdot L)$, where n is the number of variables and L is the length of the binary encoding of the ILP.*

In other words, an ILP Feasibility instance can be solved in FPT-time with respect to the number n of variables. Note that this assumes that the ILP is given explicitly; if the constraints are only given implicitly (see Section 2.3.2), writing them out may already take exponential time in the size of the original input. *Linear Programming* is a widely applied technique for modeling and solving optimization problems with linear constraints.

Definition 2.33 (Linear programs (LPs)). *A linear program (LP) is of the form:*

$$\text{max } \boldsymbol{c} \cdot \boldsymbol{x} \tag{2.4}$$

$$\text{subject to } A\boldsymbol{x} \leq \boldsymbol{b} \tag{2.5}$$

$$\boldsymbol{x} \geq 0 \tag{2.6}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)$ is an n-dimensional vector of rational variables to be determined, $\boldsymbol{c} = (c_1, \ldots, c_n)$ is a rational-valued vector, $A = (a_{ij})$ is a rational-valued $m \times n$ matrix, and $\boldsymbol{b} = (b_1, \ldots, b_m)$ is a rational-valued vector.

Feasible, optimal solutions, and *LP* Feasibility for an LP are defined analogously to the ILP case. Note that in the case of linear programming, we cannot require the variables to take integer values. Removing the integrality requirement leads to a drop in complexity, as LPs are solvable in polynomial time; see Section 2.3.2. Similar to Integer Linear Programming, Linear Programming can also solve minimization problems, as well as allow for equality ($=$) and greater than or equal ($\geq$) in its constraints.

Example 2.34. *Consider the following system of inequalities:*

$$-4x_2 \leq -1, \tag{2.7}$$

$$-8x_1 + 4x_2 \leq -1, \tag{2.8}$$

$$8x_1 + 4x_2 \leq 7. \tag{2.9}$$

If we interpret this system of inequalities as an instance of ILP Feasibility *(by adding the constraint that $x_1, x_2 \geq 0$ are integral), then it is a no-instance. Even though the system of inequalities is satisfiable, it cannot be satisfied by an integral vector $\boldsymbol{x} = (x_1, x_2)$. On the other hand, if we interpret this as an instance of LP* Feasibility*, then it is a yes-instance. The vector $\boldsymbol{x} = (0.5, 0.5)$ is an example of a feasible solution. The constraints are visualized in Figure 2.3.*

2.3.2 Ellipsoid Method

The *ellipsoid method* is an algorithm for checking the feasibility of a system of linear inequalities in polynomial time. It was originally designed for non-linear convex

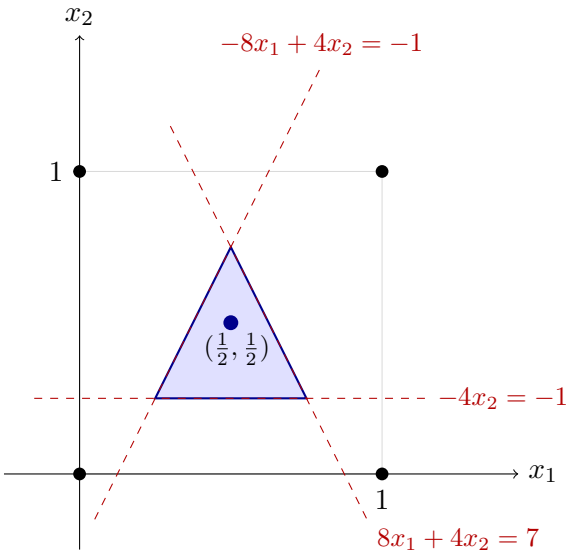


Figure 2.3: The feasible region (shaded) of the system in Example 2.34. Each dashed line is the boundary of one of the three inequalities (2.7)–(2.9); their common side is the shaded triangle. As an LP it is non-empty—for example, $(\frac{1}{2}, \frac{1}{2})$ is feasible—but it contains no integral solution, so as an ILP it is infeasible.

optimization and was later modified by Khachiyan [Kha79] to solve linear programs (see Section 2.3.1) in polynomial time. Grötschel et al. [GLS88] later extended this to solve linear programs over implicitly given constraints. The key idea is that one is not necessarily required to have a list of all inequalities. It suffices to be able to test if a vector is a solution to the system of inequalities, and if not, find a violated constraint. Such implicitly given constraints are studied in Chapter 3.

In this section, we follow Grötschel et al. [GLS88, Chapter 3] for the ellipsoid method and the more concrete presentation of Schrijver [Sch99, Chapter 14] for its application to implicitly given constraints. For further technical details, we also refer to the appendix of the book by Chvátal [Chv83].

This section aims to give an understanding of how the ellipsoid method can be used to solve LP Feasibility and implicitly given sets of linear inequalities, without going into too much technical detail.

On a very high level, the ellipsoid method is based on finding a series of ellipsoids of decreasing volume. In each step, the algorithm guarantees that if a solution exists, it must be in the current ellipsoid. The ellipsoid method then either finds a solution in the current ellipsoid or further shrinks it. If no solution is found in any of the steps, then the ellipsoid is small enough such that no solution can exist in it. This was compared by Grötschel et al. [GLS88] to the method of catching a lion in the Sahara: “Fence in the Sahara, and split it into two parts; check which part does not contain the lion, fence

the other part in, and continue. After a finite number of steps we will have caught the lion—if there was any—because the fenced-in zone will be so small that the lion cannot move anymore. Or we realize that the fenced-in zone is so small that it cannot contain any lion, i.e., there was no lion at all."

We first describe the ellipsoid method for LP Feasibility before describing the ellipsoid method for solving linear programs over implicitly given constraints. We start by defining some relevant terminology:

Definition 2.35 (Polyhedron, polytope)**.** *A* polyhedron *is a set* $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ *for a rational-valued* $m \times n$ *matrix* $A = (a_{ij})$ *and rational-valued vector* $\boldsymbol{b} = (b_1, \ldots, b_m)$*. A* polytope *is a bounded polyhedron.*

The polyhedron is the set of all points satisfying a finite set of linear inequalities. For an instance of LP Feasibility, the corresponding polyhedron P contains all feasible solutions.

Definition 2.36 (Ellipsoid)**.** *A set* $E \subseteq \mathbb{R}^n$ *is an* ellipsoid *if there exists a vector* $\boldsymbol{a} \in \mathbb{R}^n$ *and a positive definite*[5] $n \times n$ *matrix* D *such that*

$$E = E(D, \boldsymbol{a}) = \{\boldsymbol{x} \in \mathbb{R}^n \mid (\boldsymbol{x} - \boldsymbol{a})^\top D^{-1}(\boldsymbol{x} - \boldsymbol{a}) \leq 1\}.$$

The vector $\boldsymbol{a}$ *is called the center of the ellipsoid. The* volume of an ellipsoid *E is denoted by* $vol(E)$*.*

Note that the volume of an ellipsoid is only dependent on the determinant of the matrix D and the dimension of the space. As the exact computation of the volume is not of importance in this section, we refer to the aforementioned books for more details.

The ellipsoid method for LPs. As an input, we are given an instance of LP Feasibility, i.e., we ask whether the polyhedron $P = \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R}^n, A\boldsymbol{x} \leq \boldsymbol{b}\}$, where $A = (a_{ij})$ is a rational-valued $m \times n$ matrix and $\boldsymbol{b} = (b_1, \ldots, b_m)$ is a rational-valued vector, is empty or not. To keep the description simple, we make two simplifying assumptions. Neither of these assumptions is necessary for the ellipsoid method to work:

1. P is bounded, and an upper bound is given in the form of an area that must contain P (if it is non-empty), i.e., P is a polytope.
2. If P is non-empty, it is full-dimensional, i.e., it is an n-dimensional object.

The second assumption gives us a lower bound on the volume of the polytope P.

[5] An $n \times n$ matrix D is positive definite if it is symmetric, i.e., $D_{ij} = D_{ji}$ for all $i, j \in [n]$, and it satisfies $\boldsymbol{x}^\top D\boldsymbol{x} > 0$ for all non-zero $\boldsymbol{x} \in \mathbb{R}^n$.

Lemma 2.37 ([GLS88]). *If the polytope P is full-dimensional, then vol$(P) \geq 2^{-(n+1)\langle A\rangle+n^3}$, where n is the number of variables and $\langle A\rangle$ is the length of the encoding of the matrix A.*

Therefore, our assumptions ensure that we have an upper bound and a lower bound on the volume of P.

We now describe how the ellipsoid method solves an instance of LP Feasibility that satisfies these assumptions. In the following, let $E_0, \ldots, E_\ell$ be a sequence of ellipsoids. Let E_0 be the minimum-volume[6] ellipsoid that contains the given area containing P. Given an ellipsoid E_i, the method does the following: It first checks if the center c_i of the ellipsoid E_i lies in the polytope P, i.e., whether $c_i \in P$. If $c_i \in P$, then the algorithm terminates. In the case that $c_i \notin P$, the algorithm finds a violated linear constraint, i.e., it finds a row $\boldsymbol{a}_t \in A$ such that $\boldsymbol{a}_t \cdot c_i > b_t$. This violated constraint is a hyperplane through the ellipsoid E_i that cuts the ellipsoid into two "halves"[7]. By adding the constraint $\boldsymbol{a}_t \cdot \boldsymbol{x} \leq b_t$ to the ellipsoid E_i, we know which "half" of the ellipsoid needs to contain the polytope P. Around this half, we can construct the minimum-volume ellipsoid, which will be the ellipsoid E_{i+1}. It is guaranteed that $\frac{\text{vol}(E_{i+1})}{\text{vol}(E_i)} < e^{-\frac{1}{2n}}$. We repeat this process until either $c_i \in P$, or we reach an ellipsoid with volume strictly smaller than the lower bound on the volume of the polytope, see Lemma 2.37. If we reach such an ellipsoid, then it follows that the polytope is empty.

Each iteration can be computed in polynomial time in the length of the input: Determining whether $c_i \in P$, and if not, finding a violated constraint can be done by going through each of the constraints and checking if they are violated. A minimum-volume ellipsoid can also be computed in polynomial time. As we are given an upper bound as part of the input, the volume of our initial ellipsoid E_0 is bounded in terms of the length of the encoding of the instance. As the volume of the ellipsoid shrinks by a factor of at least $e^{-\frac{1}{2n}}$ per iteration, the volume decreases by a constant factor every $O(n)$ iterations. Since the logarithm of the ratio between vol(E_0) and the lower bound of Lemma 2.37 is polynomially bounded in the length of the encoding, the method terminates after polynomially many iterations. Therefore, the ellipsoid method runs in polynomial time in the length of the input.

Through the ellipsoid method, Khachiyan derived the following theorem:

Theorem 2.38 ([Kha79]). *LP Feasibility can be solved in time polynomial in the encoding length of the input.*

To solve the optimization variant with objective max $\boldsymbol{c} \cdot \boldsymbol{x}$, we can add a constraint $\boldsymbol{c} \cdot \boldsymbol{x} \geq z$ and binary search over all possible values of z.

[6] The actual minimum-volume ellipsoid cannot be computed in polynomial time, as the exact computation may require irrational roots that cannot be represented exactly. It is possible to find a $(1+\epsilon)$ approximation of the minimum-volume ellipsoid in polynomial time. Henceforth, we will refer to this approximately minimum-volume ellipsoid as the minimum-volume ellipsoid.

[7] The quotation marks are added, as the two halves need not be of the same size.

The ellipsoid method for LPs over implicitly given constraints. In the previous case, we were given an explicit LP formulation, i.e., a list of constraints defining the polytope P. However, in some cases, these constraints may not be given explicitly, as the number of constraints is large and listing them explicitly is infeasible. In these cases, the constraints can be given implicitly through some efficient encoding. We will refer to the polytope that is defined by such implicit constraints as an *implicitly given polytope*. As an example, consider the matching polytope of a graph $G = (V, E)$. The variables in the constraints correspond to the edges E in the graph. The polytope is given by the following constraints, where the variable x_e corresponds to edge $e \in E$:

$$\begin{aligned} x_e &\geq 0 && \text{for all } e \in E, \\ \textstyle\sum_{e \in E : v \in e} x_e &\leq 1 && \text{for all } v \in V, \\ \textstyle\sum_{e \in E : e \subseteq S} x_e &\leq \left\lfloor \tfrac{|S|}{2} \right\rfloor && \text{for all } S \subseteq V \text{ with } |S| \geq 3 \text{ odd.} \end{aligned}$$

For a matching $M \subseteq E$ (see Section 2.1), its incidence vector is given by the vector with $x_e = 1$ if $e \in M$ and $x_e = 0$ otherwise. The polytope is exactly the convex hull of the incidence vectors of matchings [Edm65]. The third group of constraints[8] is of exponential size. These constraints could, for example, be represented compactly using graphs (see Chapter 3 for similar examples). The key behind the ellipsoid method over implicitly given constraints is that in each iteration, we only need to determine whether the current center $c_i \in P$, or, if not, find a violated constraint. If this one step can be carried out in polynomial time, the method still runs in polynomial time. This motivates the notion of a separation oracle.

Definition 2.39 (Separation oracle)**.** *A* separation oracle *(also* separation algorithm*) is an algorithm that, given an implicitly given polytope P and a vector $\boldsymbol{x}$, determines whether $\boldsymbol{x} \in P$, and, if $\boldsymbol{x} \notin P$, returns a violated constraint, i.e., a vector $\boldsymbol{a}_t$ and a scalar b_t such that $\boldsymbol{a}_t \cdot \boldsymbol{y} \leq b_t$ for all $\boldsymbol{y} \in P$ but $\boldsymbol{a}_t \cdot \boldsymbol{x} > b_t$.*

Under the same assumptions as in the previous case, the method then proceeds as follows: It starts with a minimum-volume ellipsoid E_0 and computes a sequence of ellipsoids. Given an ellipsoid E_i, it calls the separation oracle with the center c_i as the input vector (and the implicitly defined polytope P). The separation oracle then either determines that $c_i \in P$ and the algorithm terminates, or it finds a violated constraint. The method then again splits the ellipsoid into two halves and constructs the minimum-volume ellipsoid around the half that must contain P. The algorithm again runs until either $c_i \in P$, or we reach an ellipsoid with volume strictly smaller than the lower bound on the volume of the polytope, see Lemma 2.37.

Through the ellipsoid method, the following theorem was deduced by Grötschel et al.:

Theorem 2.40 ([GLS88])**.** *Let $P \subseteq \mathbb{R}^n$ be an implicitly given polyhedron such that there exists a system of inequalities describing P, where every inequality has encoding length*

[8] Without them, setting $x_e = \frac{1}{2}$ for every edge on an odd cycle satisfies the remaining constraints, but is not part of the convex hull.

at most φ. Given a separation oracle for P, it is possible to find a vector $\boldsymbol{x} \in P$ or determine that P is empty, using a number of separation-oracle calls and arithmetic operations polynomial in n and φ. If the separation oracle runs in time polynomial in n and φ, then it is possible to find a vector $\boldsymbol{x} \in P$ or determine that P is empty, in time polynomial in n and φ.

Remark 2.41. *Although the ellipsoid method is efficient in theory, i.e., it has a polynomial upper bound on the running time, it is not efficient in practice. It performs worse in practice than algorithms like the simplex method, which, despite having exponential worst-case running time, are highly efficient on most real-world instances.*

2.4 Cooperative Games

In this section, we introduce some basic terminology needed to understand the cooperative games we study. We follow the definitions and notations from Peleg and Sudhölter [PS07]. Note that we do not consider non-cooperative games in this thesis. What differentiates cooperative games from non-cooperative games is the possibility of binding contracts. Players can agree to follow a certain strategy or share a payoff in a specific way and are bound by this agreement. This mirrors economics, where violating a contract can lead to large monetary penalties. Cooperative games give each subset of players, also called a coalition, options for dividing the value generated by the coalition.

Remark 2.42. *Throughout this thesis, we use different terms to refer to players, depending on the context. In Chapter 3, we refer to the players as agents, as the use-case is the multi-agent setting. In Chapter 4, we refer to the players as organizations, motivated by the literature. Finally, in Chapter 5, we refer to the players as voters, due to the fact that the games are based on multi-winner elections. In this section, we will refer to them as players.*

An example of a cooperative game is the so-called glove game. In this game, the players are partitioned into two groups, L and R. All players in L own a left glove and all players in R own a right glove. The market price of a pair of gloves is 1, and the value of only a left or right glove is 0. A player's glove is worthless on its own, since selling a pair requires one player from L and one from R. The goal of the game is for the players to cooperate and form contracts on how to share the resulting profit. Although in the glove game players may form any coalition, we focus only on the grand coalition, i.e., the entire set of players. In terms of the glove game, the question we study would be: "Can all the players together form a contract on how to split the total profit they can generate by cooperating?" We consider two notions to help answer this question. The first notion is individual rationality, which requires that no player could be better off on their own. In the glove game, as long as each player receives a non-negative part of the profit, the solution is individually rational. This holds because no player can generate any profit on their own. The second notion is core stability, which requires that no subset of players S could deviate such that every player in S is better off. In the glove game, this requires

that each pair of players $i_1 \in L$ and $i_2 \in R$ satisfies that they attain at least a utility of 1 together. Indeed, this is not only a necessary condition, but also a sufficient one. Every larger coalition S has a utility of $\min\{|S \cap L|, |S \cap R|\}$. Therefore, S being a blocking coalition implies that there exists a pair of players $i_1 \in L \cap S$ and $i_2 \in R \cap S$ such that they have less than 1 utility together, by the pigeonhole principle.

One way to satisfy this condition would be to give every player in L utility of 1 if $|L| < |R|$, and every player in R utility of 1, otherwise. Note that core stability implies individual rationality, as singleton subsets of players could also deviate and thereby violate core stability.

Cooperative games can be classified into two categories: those with non-transferable utilities (NTU) and those with transferable utilities (TU).

2.4.1 Games with Non-Transferable Utility (NTU Games)

Definition 2.43 (NTU game)**.** *In a non-transferable utility (NTU) game, we are given a pair* (N, μ)*, where* N *is a set of* n *players and* $\mu\colon 2^{\mathsf{N}} \setminus \{\emptyset\} \to 2^{\mathbb{R}^n}$ *is a utility function mapping every non-empty subset of players to a subset of feasible vectors in* $\mathbb{R}^n$*. A feasible outcome is a vector* $\boldsymbol{x} \in \mu(\mathsf{N})$*.*

Informally, $\mu(S)$ for a coalition $S \subseteq \mathsf{N}$ can be interpreted as the set of possible strategies for the coalition. For $i \in S$ and $\boldsymbol{x} \in \mu(S)$, $\boldsymbol{x}_i$ can be interpreted as the payoff that player i gets when choosing strategy $\boldsymbol{x}$. The value $\boldsymbol{x}_i$ is also referred to as the *utility of player* i*.*

Remark 2.44. *Equivalently to the vector notation, we can interpret the possible strategies as functions mapping the players to utilities, i.e.,* $\boldsymbol{x}(i) \coloneqq \boldsymbol{x}_i$ *for all* $i \in N$*. We sometimes use the function notation instead of the vector notation.*

We distinguish two kinds of games in this thesis:

Definition 2.45 (Maximization and minimization games (NTU))**.** *An NTU game* (N, μ) *is a maximization game if every player aims to maximize the utility they gain, i.e., player* $i \in \mathsf{N}$ *weakly prefers* $\boldsymbol{x} \in \mathbb{R}^n$ *over* $\boldsymbol{y} \in \mathbb{R}^n$ *if* $\boldsymbol{x}_i \geq \boldsymbol{y}_i$ *and (strictly) prefers* $\boldsymbol{x} \in \mathbb{R}^n$ *over* $\boldsymbol{y} \in \mathbb{R}^n$ *if* $\boldsymbol{x}_i > \boldsymbol{y}_i$*. An NTU game* (N, μ) *is a minimization game if every player aims to minimize a cost they incur, i.e., player* $i \in \mathsf{N}$ *weakly prefers* $\boldsymbol{x} \in \mathbb{R}^n$ *over* $\boldsymbol{y} \in \mathbb{R}^n$ *if* $\boldsymbol{x}_i \leq \boldsymbol{y}_i$ *and (strictly) prefers* $\boldsymbol{x} \in \mathbb{R}^n$ *over* $\boldsymbol{y} \in \mathbb{R}^n$ *if* $\boldsymbol{x}_i < \boldsymbol{y}_i$*.*

The aforementioned glove game is a maximization game, as each player aims to maximize the profit they gain. We now define several important terms:

Definition 2.46 (Individual rationality, core stability (NTU))**.** *Let* (N, μ) *be a maximization NTU game. We say that a feasible outcome* $\boldsymbol{x} \in \mu(\mathsf{N})$ *is individually rational if for every player* $i \in \mathsf{N}$ *and every* $\boldsymbol{y} \in \mu(\{i\})$ *it holds that* $\boldsymbol{x}_i \geq \boldsymbol{y}_i$*. Given a feasible outcome* $\boldsymbol{x} \in \mu(\mathsf{N})$*, we say that a coalition* $S \subseteq \mathsf{N}$ *is blocking* $\boldsymbol{x}$ *if there exists* $\boldsymbol{y} \in \mu(S)$ *such that*

$\boldsymbol{y}_i > \boldsymbol{x}_i$ holds for all $i \in S$. We say that a vector $\boldsymbol{x} \in \mathbb{R}^n$ is in the core (or core-stable) if $\boldsymbol{x} \in \mu(\mathsf{N})$ and no coalition $S \subseteq \mathsf{N}$ blocks it. A feasible outcome in the core is also referred to as a core solution or an element in the core. For minimization games, the terms are defined by flipping the inequalities.

Informally, a feasible outcome is individually rational if no player can achieve a better outcome by choosing to be on their own, and a feasible outcome is in the core if there is no subset of players who can all improve their outcomes by forming a coalition.

Example 2.47 (Teamwork at university)**.** *Consider a group project at university. There are three tasks that are equally laborious for the students: They have to research the topic, design slides, and present in front of a large crowd. Three students, 1, 2, and 3, think about whether they should collaborate on a project and, if so, how they should divide the labor. Each student's utility is given by the number of tasks that the student does not have to do. Therefore, the utility of each student is a non-negative integer in $\{0, 1, 2, 3\}$. We will write the students' utilities as vectors of the form (x, y, z), where x, y, and z are the first, second, and third students' utilities, respectively. If a student works alone, they must do all three tasks themselves and gain no relief, so $\mu(\{i\}) = \{(0, 0, 0)\}$ for $i \in [3]$. If two students form a coalition, they share the three tasks between them, so their utilities are non-negative integers summing to 3; for example, $\mu(\{1, 2\}) = \{(x, y, 0) \mid x + y = 3,\ x, y \in \mathbb{N}_0\}$, where x and y are the utilities of students 1 and 2 (student 3 is not in the coalition and gains nothing). If all three collaborate, each task done by one student relieves the other two, so the three utilities sum to 6. In total, we get the following utility function:*

$$\begin{aligned}
\mu(\{i\}) &= \{(0,0,0)\} \text{ for } i \in [3] \\
\mu(\{1,2\}) &= \{(x,y,0) \mid x+y=3, x,y \in \mathbb{N}_0\} \\
\mu(\{1,3\}) &= \{(x,0,y) \mid x+y=3, x,y \in \mathbb{N}_0\} \\
\mu(\{2,3\}) &= \{(0,x,y) \mid x+y=3, x,y \in \mathbb{N}_0\} \\
\mu(\{1,2,3\}) &= \{(x,y,z) \mid x+y+z=6, x,y,z \in \{0,1,2,3\}\}
\end{aligned}$$

Then every feasible outcome $\boldsymbol{x} \in \mu(\{1, 2, 3\})$ is individually rational, as each student on their own can achieve a utility of at most 0 anyway. Since neither singletons nor the grand coalition can block a feasible vector, we need only consider blocking coalitions of size 2. A size-2 coalition can secure a combined utility of 3 on its own, so it can block only if its members receive a combined utility below 3. Since $x + y + z = 6$ and $z \leq 3$, we have $x + y = 6 - z \geq 3$, and analogously $y + z \geq 3$ and $x + z \geq 3$. Hence, no coalition blocks, and every feasible outcome $\boldsymbol{x} \in \mu(\{1, 2, 3\})$ is core-stable.

2.4.2 Games with Transferable Utility (TU Games)

Definition 2.48 (TU game, allocation, preimputation)**.** *In a transferable utility (TU) game, we are given a pair (N, ν), where N is a set of n players and $\nu\colon 2^{\mathsf{N}} \to \mathbb{R}_{\geq 0}$ with $\nu(\emptyset) = 0$ is a characteristic function mapping every subset of players to a non-negative real number. An allocation is a vector $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n) \in \mathbb{R}^n_{\geq 0}$ that assigns to each*

player $i \in \mathsf{N}$ *a value* $\boldsymbol{\alpha}_i$*, the* utility of player i*. An allocation is called a* preimputation *if* $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = \nu(\mathsf{N})$.

Remark 2.49. *Equivalently to the vector notation, we can interpret the allocations as functions mapping the players to utilities, i.e.,* $\boldsymbol{\alpha}(i) \coloneqq \boldsymbol{\alpha}_i$ *for all* $i \in N$*. We sometimes use the function notation instead of the vector notation.*

Analogously to NTU games (see Definition 2.45), we consider two types of games:

Definition 2.50 (Maximization and minimization games (TU))**.** *A TU game* (N, ν) *is a* maximization game *if every player prefers larger values, i.e., player* $i \in \mathsf{N}$ weakly prefers $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ *over* $\boldsymbol{y} \in \mathbb{R}^n_{\geq 0}$ *if* $\boldsymbol{x}_i \geq \boldsymbol{y}_i$ *and* (strictly) prefers $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ *over* $\boldsymbol{y} \in \mathbb{R}^n_{\geq 0}$ *if* $\boldsymbol{x}_i > \boldsymbol{y}_i$*. A TU game* (N, ν) *is a* minimization game *if every player prefers smaller values, i.e., player* $i \in \mathsf{N}$ weakly prefers $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ *over* $\boldsymbol{y} \in \mathbb{R}^n_{\geq 0}$ *if* $\boldsymbol{x}_i \leq \boldsymbol{y}_i$ *and* (strictly) prefers $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ *over* $\boldsymbol{y} \in \mathbb{R}^n_{\geq 0}$ *if* $\boldsymbol{x}_i < \boldsymbol{y}_i$.

We now define several important terms for TU-games (for the analogs in the NTU setting, see Definition 2.46):

Definition 2.51 (Individual rationality, core stability (TU))**.** *Let* (N, ν) *be a maximization TU game. We say that a preimputation* $\boldsymbol{\alpha} \in \mathbb{R}^n_{\geq 0}$ *is* individually rational *if for every player* $i \in \mathsf{N}$ *it holds that* $\boldsymbol{\alpha}_i \geq \nu(\{i\})$*. An individually rational preimputation is also referred to as an* imputation*. Given a preimputation* $\boldsymbol{\alpha} \in \mathbb{R}^n_{\geq 0}$*, we say that a coalition* $S \subseteq \mathsf{N}$ *is* blocking $\boldsymbol{\alpha}$ *if* $\sum_{i \in S} \boldsymbol{\alpha}_i < \nu(S)$*. We say that a preimputation* $\boldsymbol{\alpha} \in \mathbb{R}^n_{\geq 0}$ *is in the* core (or core-stable) *if no coalition* $S \subseteq \mathsf{N}$ *blocks* $\boldsymbol{\alpha}$*. An allocation in the core is also referred to as a* core solution*, a* core allocation*, or an* element in the core*. For minimization games, the terms are defined by flipping the inequalities, i.e., a preimputation is individually rational if* $\boldsymbol{\alpha}_i \leq \nu(\{i\})$ *for every player* $i \in \mathsf{N}$*, and a coalition* $S \subseteq \mathsf{N}$ *is blocking if* $\sum_{i \in S} \boldsymbol{\alpha}_i > \nu(S)$.

Informally, an allocation is individually rational if no player is better off on their own, i.e., in a maximization game, no player can achieve a higher utility on their own, and in a minimization game, no player can achieve a lower cost on their own. An allocation is in the core if no subset of players can improve by forming a coalition.

A core element $\boldsymbol{\alpha} \in \mathbb{R}^n_{\geq 0}$ for a TU game can be modeled via an LP formulation (see Section 2.3.1). For maximization games, this takes the form:

$$\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = \nu(\mathsf{N}) \tag{2.10}$$

$$\sum_{i \in S} \boldsymbol{\alpha}_i \geq \nu(S) \quad \forall \emptyset \neq S \subseteq \mathsf{N} \tag{2.11}$$

$$\boldsymbol{\alpha}_i \geq 0 \quad \forall i \in \mathsf{N} \tag{2.12}$$

For minimization games, we replace constraint (2.11) with the following constraint:

$$\sum_{i \in S} \boldsymbol{\alpha}_i \leq \nu(S) \;\; \forall \emptyset \neq S \subseteq \mathsf{N} \tag{2.13}$$

Constraint (2.10) ensures that $\boldsymbol{\alpha}$ is a preimputation. Constraint (2.11) (respectively Constraint (2.13) for minimization games) ensures that no coalition can block. Note that for maximization games, Constraint (2.12) is not necessary, as it is implied by Constraint (2.11) and the fact that ν takes non-negative values, but it is added for the sake of completeness. For minimization games, Constraint (2.12) is not implied by the remaining constraints. Note that the existence of this LP formulation does not mean that a core allocation can be found in polynomial time with respect to the length of the encoding of (N, ν). Constraint (2.11) and Constraint (2.13) consist of an exponential (in the number of agents) number of constraints that could have a compact encoding. A maximization TU game is called *monotonic* if for all $S, T \subseteq \mathsf{N}$ such that $S \subseteq T$, it holds that $\nu(S) \leq \nu(T)$. Similarly, a minimization TU game is called monotonic if for all $S, T \subseteq \mathsf{N}$ such that $S \subseteq T$, it holds that $\nu(S) \geq \nu(T)$. Informally, this means that adding players to a coalition cannot put the coalition in a worse position.

Remark 2.52. *NTU games are a generalization of TU games. Each TU game can be formulated as an NTU game, by defining the feasible vectors for each coalition $\emptyset \neq S \subseteq \mathsf{N}$ as $\mu(S) \coloneqq \{\boldsymbol{\alpha} \mid \sum_{i \in S} \boldsymbol{\alpha}_i = \nu(S), \boldsymbol{\alpha}_j = 0, \forall\, j \notin S\}$. Intuitively, in TU games, all ways of dividing the utility among the players in the coalition are feasible outcomes, whereas NTU games can have more general or more restricted outcomes.*

Example 2.53 (Teamwork at university)**.** *Consider the situation from Example 2.47. The students have now realized that, rather than only dividing tasks among themselves, they can also choose to work on tasks together, thereby splitting the total workload for each task. The students still derive their utility by how much of the total workload they do not have to do, but now they can choose to split tasks as they see fit. This gives us the following characteristic function:*

$$\nu(S) = 0, \text{ if } |S| = 1$$
$$\nu(S) = 3, \text{ if } |S| = 2$$
$$\nu(S) = 6, \text{ if } |S| = 3$$

Then every preimputation $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}_i \geq 0$ for all $i \in [3]$ is individually rational, since a lone student completes every task themselves and so has value $\nu(\{i\}) = 0$. Recall that a coalition S blocks if its members are jointly awarded less than they could secure by themselves, i.e., $\sum_{i \in S} \boldsymbol{\alpha}_i < \nu(S)$. Singletons have $\nu(\{i\}) = 0$ and so never block, and the grand coalition must satisfy the preimputation condition $x + y + z = 6 = \nu(\{1, 2, 3\})$. The only coalitions that can block are the three coalitions of size 2, each with $\nu(S) = 3$. The core is therefore $\{(x, y, z) \mid x + y + z = 6,\ x + y \geq 3,\ y + z \geq 3,\ x + z \geq 3,\ x, y, z \in \mathbb{R}_{\geq 0}\}$, for example the equal split $(2, 2, 2)$. As we are in a TU game, any real allocation satisfying the conditions—e.g., $(2.5, 1.5, 2)$—is also in the core. This was not the case in the NTU counterpart.

2.5 Scheduling

Scheduling is a field that deals with assigning *jobs* with processing times to resources, which we call *machines*, and fixing when each job runs, to optimize an objective. We introduce only the concepts used in this thesis and refer the reader to the book by Pinedo [Pin22] for a more comprehensive overview.

We only consider the *parallel identical machines model*[9], i.e., a model in which m identical machines can process jobs simultaneously. Each machine can process only one job at a time, and each job can be processed by only one machine. Jobs are non-preemptive, i.e., every machine has to complete a job before processing another, and all jobs are released at time zero.

An instance is given by a tuple $(\mathcal{M}, \mathcal{J}, (\mathsf{p}_j)_{j\in[n]})$, where

- $\mathcal{M}$ is a set of m identical machines,
- $\mathcal{J} = \{\alpha_1, \dots, \alpha_n\}$ is a set of n jobs and
- processing times $(\mathsf{p}_j)_{j\in[n]} \in \mathbb{N}$, where p_j is the processing time of job α_j.

Note that a job's processing time does not depend on the machine.

Definition 2.54. *A schedule $\sigma\colon \mathcal{J} \to \mathcal{M} \times \mathbb{N}$ is a function that assigns to each job a machine and a completion time. We denote by $\mathsf{m}_j(\sigma)$ the scheduled machine and by $\mathsf{C}_j(\sigma)$ the scheduled completion time of the j-th job α_j under schedule σ. A schedule σ is feasible if*

- *for each job $\alpha_j \in \mathcal{J}$, $\mathsf{C}_j(\sigma) \geq \mathsf{p}_j$ and*
- *for each pair of jobs $\alpha_j, \alpha_{j'}$ scheduled on the same machine with $\mathsf{C}_j(\sigma) \geq \mathsf{C}_{j'}(\sigma)$, it holds that $\mathsf{C}_j(\sigma) - \mathsf{C}_{j'}(\sigma) \geq \mathsf{p}_j$.*

Informally, a schedule is feasible if each job completes no earlier than its processing time (the first requirement) and no two jobs are processed on the same machine simultaneously (the second requirement). We will mostly use "schedules" to refer to "feasible schedules".

In this thesis, we consider two optimization criteria: the *makespan ($\mathsf{C}_{\max}$)* and the *sum of completion times (ΣC)*. We abbreviate the sum of completion times as *Σ-time*.

Definition 2.55 (Makespan, sum of completion times)**.** *Let σ be a schedule. Then, the makespan (resp. the sum of completion times) of a set of jobs $\mathcal{J}' \subseteq \mathcal{J}$ with respect to σ, denoted as $\mathsf{C}_{\max}(\sigma, \mathcal{J}')$ (resp. $\mathsf{C}_{\Sigma}(\sigma, \mathcal{J}')$), is the maximum completion time (resp. sum of completion times) of all jobs in $\mathcal{J}'$, i.e., $\mathsf{C}_{\max}(\sigma, \mathcal{J}') = \max_{\alpha_j \in \mathcal{J}'} \mathsf{C}_j(\sigma)$ and $\mathsf{C}_{\Sigma}(\sigma, \mathcal{J}') = \sum_{\alpha_j \in \mathcal{J}'} \mathsf{C}_j(\sigma)$.*

[9] The scheduling community refers to this problem as the $P\,||\cdot$ environment in the standard three-field scheduling notation.

2.6 Multi-Winner Approval Voting

In this section, we introduce some voting rules and basic definitions from multi-winner approval voting. For a more comprehensive treatment, we refer to Lackner and Skowron [LS23]. For an overview of parameterized complexity results in multi-winner elections, we refer to the survey paper by Chen et al. [CHS26]. An *approval preference profile* (or *profile* for short) is a triple $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$, where $\mathcal{C}$ denotes a set of m alternatives with $\mathcal{C} = \{c_1, \dots, c_m\}$, $\mathcal{V}$ a set of n voters with $\mathcal{V} = \{v_1, \dots, v_n\}$, and a collection $\mathcal{R} = (\mathsf{A}_1, \dots, \mathsf{A}_n)$ of subsets (*approval ballots*) of $\mathcal{C}$, with A_i[10] consisting of the alternatives that voter v_i approves of, $i \in [n]$. For each alternative $c_j \in \mathcal{C}$, we use $\mathsf{V}(c_j)$ to denote the set of voters who approve of c_j. We say that two alternatives a and b are of the same *type* if they are approved by the same subset of voters, i.e., $\mathsf{V}(a) = \mathsf{V}(b)$. Similarly, we say that two voters v and w are of the same *type* if they approve the same subset of alternatives.

An instance of *multi-winner approval voting* (also called *multi-winner approval election*) is a pair $(\mathcal{P}, k)$, where $\mathcal{P}$ denotes an approval preference profile and $k \in [m]$ denotes the size of a committee, i.e., the number of alternatives from $\mathcal{C}$ that shall be selected. The task is to select a *committee*, i.e., a subset of alternatives from $\mathcal{C}$, of size k. The committee is selected using so-called *voting rules*. We focus on score-based voting rules. A *score-based (multi-winner) voting rule* λ (*voting rule* for short) takes as input an instance $(\mathcal{P}, k)$ of multi-winner approval voting, and assigns to each committee W a score $\mathrm{sc}_\lambda(\mathcal{P}, W)$. The committees with maximum score are the *winning committees*. The score $\mathrm{sc}_\lambda(\mathcal{P}, W)$ measures the satisfaction of the voters towards committee W. We omit the profile $\mathcal{P}$ and the subscript λ if they are clear from the context. Note that usually, λ selects a unique winning committee by employing tie-breaking among the winning committees.

In this thesis, we consider four score-based voting rules: approval voting (AV), satisfaction approval voting (SAV), the Chamberlin–Courant rule (CC), and proportional approval voting (PAV). For each rule $\lambda \in \{\mathrm{AV}, \mathrm{SAV}, \mathrm{CC}, \mathrm{PAV}\}$, the score of a committee is the sum of the voters' individual scores. Therefore, we will define the score of a committee for each subset $\mathcal{V}' \subseteq \mathcal{V}$, as $\mathsf{sc}_\lambda(\mathcal{V}', W)$. It then holds that $\mathrm{sc}_\lambda(\mathcal{P}, W) = \mathsf{sc}_\lambda(\mathcal{V}, W)$. Formally, given a subset $\mathcal{V}' \subseteq \mathcal{V}$ of voters and a committee $W \subseteq \mathcal{C}$, define:

Approval Voting (AV): $\mathsf{sc}_{AV}(\mathcal{V}', W) = \sum_{v_i \in \mathcal{V}'} |\mathsf{A}_i \cap W|$.

Informally, the AV rule selects a size-k committee that maximizes the sum of approving voters.

Satisfaction Approval Voting (SAV): $\mathsf{sc}_{SAV}(\mathcal{V}', W) = \begin{cases} \sum_{v_i \in \mathcal{V}'} \frac{|\mathsf{A}_i \cap W|}{|\mathsf{A}_i|} & \mathsf{A}_i \neq \emptyset \\ 0 & \text{else} \end{cases}$.

Informally, the SAV rule selects a size-k committee that maximizes the weighted sum of approving voters, where the weight of a voter is inversely proportional to the number of the voter's approved alternatives.

[10] A_i can be empty.

Chamberlin–Courant (CC): $\mathsf{sc}_{CC}(\mathcal{V}', W) = \sum_{v_i \in \mathcal{V}'} \min(|\mathsf{A}_i \cap W|, 1)$.

Informally, the CC rule selects a size-k committee that maximizes the number of voters who approve of at least one alternative in the committee.

Proportional Approval Voting (PAV): $\mathsf{sc}_{PAV}(\mathcal{V}', W) = \sum_{v_i \in \mathcal{V}'} \sum_{z=1}^{|\mathsf{A}_i \cap W|} \frac{1}{z}$.

Informally, the PAV rule selects a size-k committee that maximizes the sum of voters' satisfaction, where a voter who approves j alternatives in the committee contributes the j-th harmonic number $h(j) = \sum_{i=1}^{j} \frac{1}{i}$. This voting rule models that each additional approved alternative in the committee adds less to the satisfaction than the one before.

For readability, we write $\mathsf{sc}_\lambda(v_i, W)$ instead of $\mathsf{sc}_\lambda(\mathcal{V}', W)$ when $\mathcal{V}' = \{v_i\}$. We introduce a general class of voting rules that contains several of the previously mentioned voting rules.

Definition 2.56 (Thiele methods)**.** *A voting rule λ is a Thiele method if there exists a non-decreasing function $\omega\colon \mathbb{N}_0 \to \mathbb{R}_{\geq 0}$ with $\omega(0) = 0$, such that $\mathsf{sc}_\lambda(\mathcal{V}, W) = \sum_{v_i \in \mathcal{V}} \omega(|\mathsf{A}_i \cap W|)$.*

Through the definitions of the voting rules, we can directly observe the following:

Observation 2.57. *AV, CC, and PAV are Thiele methods. SAV is not a Thiele method.*

Remark 2.58. *Computing a winning committee under the CC rule is equivalent to solving the Maximum Coverage (Max Cover) problem, in which one is given a universe $\mathcal{U}$, a collection of subsets $S_1, \ldots, S_m \subseteq \mathcal{U}$, and a non-negative integer k, and the goal is to select k of the subsets to cover as many elements of $\mathcal{U}$ as possible. This follows as the voters correspond to the universe $\mathcal{U}$, and each alternative corresponds to the subset of voters that approve it. A size-k winning committee then corresponds to k subsets that cover the maximum number of elements (i.e., alternatives that cover the maximum number of voters).*

Example 2.59. *Consider the following approval profile $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$, with $\mathcal{C} = \{c_1, c_2, c_3\}$, $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$, and*

$$\mathsf{A}_1 = \mathsf{A}_2 = \{c_1\},\ \mathsf{A}_3 = \{c_1, c_2\},\ \mathsf{A}_4 = \{c_2, c_3\}.$$

We can also represent the preference profile as a binary matrix. In this matrix, the rows correspond to voters and the columns to alternatives. If the table entry is a 1, then the voter corresponding to the row approves of the alternative corresponding to the column. Otherwise, the voter does not approve.

	c_1	c_2	c_3
v_1	1	0	0
v_2	1	0	0
v_3	1	1	0
v_4	0	1	1

Let $k = 2$.

AV: For AV, we can greedily pick alternatives that are approved by the most voters. The most approved alternative is c_1, which is approved by three voters, and the second most approved alternative is c_2, which is approved by two voters. This gives us $W_1 = \{c_1, c_2\}$ as the unique winning committee with score $sc_{AV}(W_1) = 5$.

CC: The winning committees are W_1 and $W_2 = \{c_1, c_3\}$, each with score $sc_{CC}(W_1) = sc_{CC}(W_2) = 4$. This can be observed, as c_1 needs to be in every winning committee; otherwise, the score is at most 2. Finally, picking either c_2 or c_3 as the second alternative leads to the same score.

PAV: W_1 is the unique winning committee with score $sc_{PAV}(W_1) = 4.5$. This follows, as W_2 has score $sc_{PAV}(W_2) = 4$, and $W_3 = \{c_2, c_3\}$ has score $sc_{PAV}(W_3) = 2.5$.

SAV: W_1 is the unique winning committee with score $sc_{SAV}(W_1) = 3.5$, as W_2 has score $sc_{SAV}(W_2) = 3$ and W_3 has score $sc_{SAV}(W_3) = 1.5$.

2.7 Problem Compendium Organized by Complexity

In this section, we provide a compendium of computational problems. The problems are categorized by complexity. For each problem, we provide some additional information, such as where the problem was proven to be complete for the class and its parameterized complexity with respect to some relevant parameters.

2.7.1 NP-Complete Problems

Vertex Cover
Input: A graph $G = (V, E)$ and a non-negative integer k.
Question: Does there exist a vertex cover, i.e., a subset $V' \subseteq V$ of vertices such that $e \cap V' \neq \emptyset$ for all $e \in E$, of size at most k?

Notes: The problem was shown to be NP-complete by Karp [Kar72] and it remains NP-hard even if all vertices have degree at most 3 [GJS76]. The problem parameterized by the solution size k is FPT [BG93].

Dominating Set
Input: A graph $G = (V, E)$ and a non-negative integer k.
Question: Does there exist a dominating set, i.e., a subset $V' \subseteq V$ of vertices such that $N_G[v] \cap V' \neq \emptyset$ for all $v \in V$, of size at most k?

Notes: The problem is NP-complete [GJ79]. The problem is W[2]-complete with respect to the solution size k [DF95a].

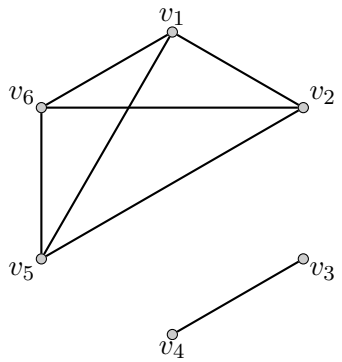


Figure 2.4: The graph $G = (V, E)$ used in Examples 2.60 and 2.66. The vertices $\{v_1, v_2, v_5, v_6\}$ form a clique of size 4, while v_3 and v_4 form an isolated edge.

Independent Set
Input: A graph $G = (V, E)$ and a non-negative integer k.
Question: Does there exist an independent set, i.e., a subset $V' \subseteq V$ of vertices such that no two vertices in V' are adjacent, of size at least k?

Notes: The problem is NP-complete and remains NP-hard if all vertices have degree at most 3 [GJ79]. The problem is W[1]-complete with respect to the solution size k [DF95b].

Example 2.60 (Vertex Cover/Dominating Set/Independent Set)**.** *Consider the graph G shown in Figure 2.4 and let $k = 2$.*

- *The corresponding Vertex Cover instance is a no-instance. This holds, as at least one vertex out of $\{v_3, v_4\}$ needs to be in the vertex cover. As there is a clique of size 4 $\{v_1, v_2, v_5, v_6\}$, at least three of those vertices need to be in a vertex cover. Therefore, a minimum-size vertex cover has size at least 4, and since $k = 2$, the given instance is a no-instance.*

- *The corresponding Dominating Set instance is a yes-instance, as $\{v_1, v_3\}$ is a dominating set of size 2. Note that it is also a minimum dominating set, i.e., a dominating set of minimum size.*

- *The corresponding Independent Set instance is a yes-instance, as $\{v_1, v_3\}$ is an independent set of size 2. Note that it is also a maximum independent set, i.e., an independent set of maximum size.*

ILP Feasibility
Input: An ILP (see Section 2.3.1).
Question: Does the ILP admit a feasible solution?

Notes: The problem is NP-complete [Pap81]. It is FPT with respect to the number of variables n [Len83, Kan87, FT87].

3-Partition
Input: A non-negative integer B, a set X of $3q$ non-negative integers $\{x_1, \ldots, x_{3q}\}$ such that $\sum_{i \in [3q]} x_i = qB = Q$.
Question: Is there a partition of X into q triples $\{T_1, \ldots, T_q\}$ such that the sum of the integers in each triple is exactly B?

Notes: The problem was shown to be NP-complete by Garey and Johnson [GJ75]. The problem remains strongly NP-hard even if all integers in X are strictly larger than $B/4$ and strictly smaller than $B/2$ [GJ75].

Example 2.61 (3-Partition)**.** *Consider the instance with $B = 10$, $q = 2$, and $X = \{3, 3, 4, 3, 3, 4\}$ such that $\sum_{i \in [6]} x_i = 20 = qB$. All integers are strictly larger than $B/4 = 2.5$ and strictly smaller than $B/2 = 5$. This is a yes-instance, as we can partition X into the triples $\{3, 3, 4\}$ and $\{3, 3, 4\}$, each summing to exactly $B = 10$.*

Unary-BinPacking
Input: A set X of y non-negative integers $\{x_1, \ldots, x_y\}$, a non-negative integer B, a non-negative integer m.
Question: Is there a partition of X into m sets $\{S_1, \ldots, S_m\}$ such that the sum of the integers in each set is at most B?

Notes: The problem is NP-complete [GJ79]. Note that the weights are encoded in unary, so the problem is strongly NP-hard. The Unary-BinPacking problem is W[1]-hard with respect to the number m of bins [JKMS13].

Example 2.62 (Unary-BinPacking)**.** *Consider the instance with $X = \{4, 2, 2, 2\}$, $B = 5$, and $m = 3$. This is a yes-instance: The items can be packed into the three sets $\{4\}$, $\{2, 2\}$, and $\{2\}$, whose sums 4, 4, and 2 are each at most $B = 5$. With only $m = 2$ bins, the same item set would be a no-instance, as the set containing 4 cannot contain another element, as then the sum would exceed 5, and the other elements sum up to 6.*

SET COVER
Input: A universe $U = [\hat{n}]$, a collection $\mathcal{S} = \{S_1, \ldots, S_{\hat{m}}\}$ with $S_j \subseteq U$ for all $j \in [\hat{m}]$, and a non-negative integer h.
Question: Is there a subcollection of size h whose union is U?

Notes: The problem was shown to be NP-complete by Karp [Kar72]. The problem is W[2]-hard with respect to the solution size h [DF95a].

RESTRICTED EXACT COVER BY 3-SETS (RX3C)
Input: A universe of $3\hat{n}$ elements $X = \{x_1, \ldots, x_{3\hat{n}}\}$, a collection $\mathcal{S} = \{S_1, \ldots, S_{\hat{m}}\}$ with $S_j \subseteq X$ and $|S_j| = 3$ for all $j \in [\hat{m}]$ such that every element appears in exactly three sets in $\mathcal{S}$, i.e., $\hat{m} = 3\hat{n}$.
Question: Does $(X, \mathcal{S})$ admit an *exact cover* (i.e., a subcollection $\mathcal{S}'$ of cardinality $\hat{n}$ whose union is X)?

Notes: This problem was shown to be NP-complete by Gonzalez [Gon85]. It is a restriction of the SET COVER problem.

Example 2.63 (SET COVER/RX3C)**.** *Consider the SET COVER instance with* $U = \{1, 2, 3, 4, 5, 6\}$ *and the collection* $\mathcal{S} = \{\{1, 2, 3\}, \{4, 5, 6\}, \{1, 2, 4\}, \{3, 5, 6\}, \{1, 4, 5\}, \{2, 3, 6\}\}$ *and* $h = 2$*. This is also an RX3C instance with* $\hat{n} = 2$*, as each set contains three elements and each element appears in exactly three sets. The given instance is a yes-instance (for both problems), as the subcollection* $\mathcal{S}' = \{\{1, 2, 3\}, \{4, 5, 6\}\}$ *satisfies that its union is* U*.*

BALANCED COMPLETE BIPARTITE SUBGRAPH (BICLIQUE)
Input: A graph $G = (U, E)$ and a non-negative integer h.
Question: Do there exist two disjoint subsets $V_1, V_2 \subseteq U$ with $|V_1| = |V_2| = h$ such that $\{u, w\} \in E$ for all $u \in V_1$ and all $w \in V_2$?

Notes: The problem was shown to be NP-complete by Garey and Johnson under the name BALANCED COMPLETE BIPARTITE SUBGRAPH [GJ79]. They showed NP-hardness even for the case where the underlying graph G is bipartite. The problem is W[1]-hard with respect to h [Lin18].

Example 2.64 (BALANCED COMPLETE BIPARTITE SUBGRAPH)**.** *Consider the instance of BALANCED COMPLETE BIPARTITE SUBGRAPH given by the graph* G *in Figure 2.5 and* $h = 2$*. This is a yes-instance, as choosing* $V_1 = \{a_1, a_2\}$ *and* $V_2 = \{b_1, b_2\}$ *gives two disjoint vertex sets of size* 2 *with* $\{u, w\} \in E$ *for all* $u \in V_1$ *and all* $w \in V_2$*, i.e., the complete bipartite subgraph* $K_{2,2}$*.*

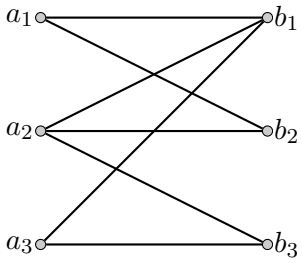


Figure 2.5: The bipartite graph $G = (U, E)$ used in Example 2.64, with parts $\{a_1, a_2, a_3\}$ and $\{b_1, b_2, b_3\}$. The four edges among $\{a_1, a_2\}$ and $\{b_1, b_2\}$ form a complete bipartite subgraph $K_{2,2}$.

2.7.2 DP-Complete Problems

SAT-UNSAT
Input: Two CNF formulas φ_1 and φ_2.
Question: Is φ_1 satisfiable and φ_2 unsatisfiable?

Notes: The problem was shown to be DP-complete by Papadimitriou and Yannakakis [PY84].

Example 2.65 (SAT-UNSAT)**.** *Consider the following SAT-UNSAT instance* (φ_1, φ_2)*, where*

- $\varphi_1 \coloneqq (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$ *and*
- $\varphi_2 \coloneqq x_3 \wedge \neg x_3$*.*

The instance is a yes-instance, as φ_1 *is satisfiable, e.g., by setting* x_1 *to True and* x_2 *to False, and* φ_2 *is unsatisfiable, as it is not possible for* x_3 *to be simultaneously True and False.*

2.7.3 Θ_2^{P}-Complete Problems

Vertex Cover Membership
Input: A graph $G = (V, E)$ and a vertex $v' \in V$.
Question: Does G admit a minimum-size vertex cover V' such that $v' \in V'$?

Notes: The problem was shown to be Θ_2^{P}-complete by Hemaspaandra et al. [HSV05].

Example 2.66 (Vertex Cover Membership)**.** *Consider the graph in Figure 2.4 again and let* $v' = v_6$*. Then the given instance is a yes-instance, as* $\{v_1, v_2, v_3, v_6\}$ *is a*

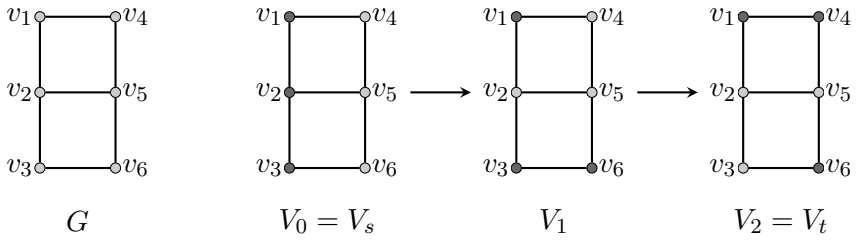


Figure 2.6: The graph $G = (V, E)$ (a 3×2 grid, left), together with a reconfiguration sequence of size-3 dominating sets from $V_s = \{v_1, v_2, v_3\}$ to $V_t = \{v_1, v_4, v_6\}$ for the DSR-TJ instance in Example 2.67. In each step exactly one vertex is replaced by another; vertices in the current dominating set are colored darker.

minimum-size vertex cover containing $v' = v_6$. Note that no vertex cover can have size less than 4, as $\{v_1, v_2, v_5, v_6\}$ is a size-4 clique and at least one of the vertices $\{v_3, v_4\}$ must be in a vertex cover.

2.7.4 PSPACE-Complete Problems

The problems in this subsection are *reconfiguration problems*. Informally, we ask if we can reconfigure one solution into another in small, feasible steps. In reconfiguration problems, we are given two feasible solutions to an underlying problem and ask whether one solution can be transformed into the other through a sequence of *adjacent solutions*. Each solution in the sequence must be a feasible solution to the underlying problem, and consecutive solutions must be adjacent. What it means for solutions to be adjacent depends on the setting. All the problems we consider use the *token jumping* rule, in which each step moves a single element (a *token*) from one vertex to another. In other words, the symmetric difference between two adjacent solutions must be at most 2. We refer to the survey of Nishimura [Nis18] for an overview and other settings.

Independent Set Reconfiguration via Token Jumping (ISR-TJ)
Input: A graph G, two non-negative integers h and $\hat{\ell}$, and two independent sets V_s, V_t of size h.
Question: Does there exist a sequence $V_0, V_1, \ldots, V_{\hat{\ell}}$ of size-h independent sets of G such that $V_0 = V_s$, $V_{\hat{\ell}} = V_t$, and $|V_i \setminus V_{i+1}| = |V_{i+1} \setminus V_i| \leq 1$ holds for all $0 \leq i \leq \hat{\ell} - 1$?

Notes: The problem was shown to be PSPACE-complete and W[1]-hard with respect to the number h of tokens by Ito et al. [IKO+14].

Dominating Set Reconfiguration via Token Jumping (DSR-TJ)
Input: A graph $G = (V, E)$, two non-negative integers h and $\hat{\ell}$, and two size-h dominating sets V_s and V_t of G.
Question: Does there exist a sequence $V_0, \ldots, V_{\hat{\ell}}$ of size-h dominating sets such that $V_0 = V_s$, $V_{\hat{\ell}} = V_t$, and $|V_i \setminus V_{i+1}| = |V_{i+1} \setminus V_i| \leq 1$ for all $0 \leq i \leq \hat{\ell} - 1$?

Notes: The problem was shown to be PSPACE-complete by Haddadan et al. [HIM+16].

Example 2.67 (ISR-TJ, DSR-TJ)**.** *Consider the graph G shown in Figure 2.6.*

- *The graph G with $V_s = \{v_1, v_3, v_5\}$ and $V_t = \{v_2, v_4, v_6\}$, $h = 3$, and arbitrary $\hat{\ell}$ is a no-instance of ISR-TJ. This can be observed, as no vertex in either independent set can be replaced with another vertex without losing the independent set property.*
- *The graph G with $V_s = \{v_1, v_2, v_3\}$ and $V_t = \{v_1, v_4, v_6\}$, $h = 3$, and $\hat{\ell} \geq 2$ is a yes-instance of DSR-TJ. A valid sequence of size-3 dominating sets is shown in Figure 2.6.*

CHAPTER 3

Partitioned Combinatorial Optimization Games

3.1 Introduction

As outlined in Chapter 1, a central theme of this thesis is to find stable cooperative agreements to incentivize cooperation; this chapter studies one such setting, where an underlying combinatorial problem describes the cost or utility of a coalition (as seen in the lecture-hall scenario in Chapter 1). To model such settings, we introduce Partitioned Combinatorial Optimization Games, a generalization of Combinatorial Optimization Games. *Combinatorial Optimization Games* (*COGs*) [BHH01, Den09] form a rich class of TU games, defined on combinatorial structures such as graphs. In COGs, we are given a combinatorial structure—such as a graph—such that each agent owns a single unit of the structure—typically a vertex or an edge—and the value of each coalition of agents is determined by the optimal solution of a combinatorial optimization problem restricted to the substructure induced by the agents in the coalition. COGs yielded several influential game-theoretic models, including *vertex cover game* (where the agents are the edges) [DIN99], *dominating set game* [vV04], *minimum cost spanning tree game* [CK73], *maximum matching game* [DIN99], and *traveling salesperson game* [FK93][1], among others. For example, in a maximum matching game, each agent is a vertex in an undirected graph such that the utility of any coalition of agents is the size of a maximum matching in the subgraph induced by the coalition.

The restriction that each agent owns exactly one element, however, limits the applicability of COGs to many contemporary collaborative scenarios where participants control

[1] Here, each agent owns a vertex in an edge-weighted graph, and there exists a distinguished vertex. The cost of any coalition is given by the shortest closed tour that starts at the distinguished vertex, visits every vertex owned by an agent in the coalition, and returns to the distinguished vertex.

multiple resources. In international multi-organization collaborations such as scheduling [PRT09] (see also Chapter 4) and international kidney exchange programs [BKPP19], a single organization may contribute dozens or even hundreds of primitive elements—e.g., cluster servers or patient–donor pairs. These *multi-ownership* settings violate the classical assumption, creating fresh challenges for fairness, stability, and computational tractability that this chapter seeks to address.

To address the limitation imposed by each agent owning exactly a single element of the combinatorial structure, we propose *Partitioned Combinatorial Optimization Games* (*PCOGs*) as a general framework extending COGs. In a PCOG, each agent owns a part of the given structure (e.g., a subset of vertices or edges), and the value of a coalition is determined by the optimal solution over the combined substructure possessed by its members. Clearly, COG is a special case of PCOG in which either all agents each own exactly one vertex or all agents each own exactly one edge. Moreover, since the coalition values are derived from the underlying combinatorial optimization problem, the input of a PCOG remains compact—its size is polynomial in the number of agents—analogously to traditional COGs. By extending the framework of COGs to PCOGs, we offer a flexible framework capable of modeling several situations in which self-interested parties, each owning several resources, are encouraged to cooperate by an outcome that no subgroup can profitably deviate from.

To model this, we utilize the notion of core stability (cf. Definition 2.51), a central solution concept in cooperative game theory that captures stable utility distributions where no coalition has an incentive to deviate. We investigate two fundamental computational problems:

1. Core-Stability Verification (CV): Is a given allocation core-stable (i.e., in the core)?
2. Core-Stability Existence (CE): Does there exist any core-stable allocation?

To gauge the computational effort required to address these problems, we analyze four classical combinatorial optimization problems on graphs: Minimum Vertex Cover (MinVC), Minimum Dominating Set (MinDS), Minimum Cost Spanning Tree (MinST), and Maximum Matching (MaxMatch). This selection allows us to systematically compare the computational impact of partitioning between NP-hard problems (the first two, see Section 2.7) and polynomial-time solvable problems (the latter two, as shown by Prim [Pri57] and Edmonds [Edm65], respectively).

Additionally, the four problems can be used to model real-world multi-organizational applications. The PCOG-MaxMatch problem models the international kidney exchange programs mentioned above.

PCOG-MinDS can be used to model the *location set covering problem* of Toregas et al. [TSRB71], a model for placing emergency facilities subject to a maximum response distance. In our model the vertices can be interpreted as demand sites and a facility

placed at a site serves it together with its neighborhood. The game becomes partitioned as soon as the demand sites (and therefore possible facility locations) are divided among multiple organizations. This is the case for shared emergency-service coverage across neighboring municipalities, as in Austria, where providing the local ambulance service is a municipal responsibility[2]. Generally, municipalities contribute sites in their own territory, but their stations are also used to serve demand in nearby municipalities. In Lower Austria the municipalities then divide the cost of the emergency medical services in the state among themselves using a fixed formula based on population size and fiscal capacity[3].

A possible use case of PCOG-MinVC would be to model placing surveillance cameras at the intersections of a road network. In this setting one entity could be responsible for multiple roads (once again one might consider municipalities as for PCOG-MinDS). A related game was studied by Gusev [Gus20]. His game is unpartitioned and, unlike the vertex cover game of Deng et al. [DIN99], has the vertices as players: A coalition has utility 1 if the corresponding vertices form a vertex cover. He used the game to determine the value of each intersection.

When COG-MinST was introduced, multiple possible interpretations of the game were proposed [CK73, Bir76]. All of them amount to the problem of sharing the cost of being connected to a common supplier: The service in question may be a telephone network, a water supply, power, or something else entirely. The partitioned version PCOG-MinST models settings where entities are responsible for connecting multiple nodes to the supplier. For example, a town may be responsible for connecting multiple households to the water supply, or an organization may own several factories that need to be connected to the power grid.

These examples also show what is actually transferred in each of these readings. For PCOG-MinVC, PCOG-MinDS, and PCOG-MinST, the quantity being divided is money. This can either be interpreted as determining a fair way to split the cost of a joint project, or as side payments being made between the agents. For example, one town pays another to use its ambulances. For PCOG-MaxMatch, the transfer can be realized through a credit system, where each country is assigned a target number of transplants for a round and the deviation from that target carries forward as credit. Such a system was discussed by Benedek et al. [BBPY24].

Main contributions. We study a generalized framework for combinatorial optimization games that naturally accommodates scenarios involving multi-organization collaboration. Our contributions are summarized as follows:

- We propose the PCOG framework and establish a *sufficient* condition for when a core-stable allocation always exists.

[2]See Art. 118 Abs. 3 Z 7 B-VG. How the service is organized is left to the nine federal states.
[3]See § 10 Abs. 2 NÖ Rettungsdienstgesetz 2017.

- **CV:** We show that for the two NP-hard optimization problems—MinDS and MinVC—the CV problem is DP-complete; see Definition 2.17 for more information on the complexity class DP. We also use a known result by Faigle et al. [FKFH97] for COG to derive coNP-completeness for PCOG-MinST. We remark that for PCOG-MaxMatch, the CV problem is known to be coNP-complete [BKPP19].

- **CE:** For PCOG-MinVC and PCOG-MinDS we establish that the core-stability existence problem CE is Θ_2^{P}-complete (see Definition 2.18 for the definition of Θ_2^{P}). While the Θ_2^{P}-hardness is relatively straightforward, the key step in the upper-bound argument is to adapt the ellipsoid method (cf. Section 2.3.2) to an auxiliary problem and show that a separation routine for the auxiliary problem lies in coNP. Our construction is inspired by the ellipsoid-based approach of Biró et al. [BKPW18, BKPP19], who obtained a straightforward *polynomial-time* algorithm for a more permissive variant of PCOG-MaxMatch in which each agent controls at most two vertices. In our setting, the underlying verification problem is already intractable, so the oracle must be engineered much more carefully. We believe that the resulting technique can also be applied to other combinatorial optimization games.

  A further interesting distinction emerges within the polynomial-time solvable cases. For PCOG-MinST, we derive from the results for COG [Bir76, GH81] that a core-stable allocation always exists and finding one can be done in polynomial time. In contrast, we complement the coNP-hardness for PCOG-MaxMatch from the literature [BKPP19] by showing its coNP-containment.

- We investigate parameterized complexity through two structural parameters: number $\hat{n}$ of agents and maximum number c_{max} of vertices (or edges) owned by any agent.

In summary, our results demonstrate that the flexibility of allowing an agent to possess a larger portion of the combinatorial structure has a more significant computational impact on NP-hard optimization problems than on those that are polynomial-time solvable. See Table 3.1 for an overview of the complexity results.

Related work. COG has been studied extensively across numerous graph optimization problems such as Vertex Cover [FKZ08, DIN99], Minimum Dominating Set [vV04, FK05], Independent Set [XWF21], Minimum Cost Spanning Tree [CK73, Bir76, GH81, FKFH97], Maximum Matching [DIN99, GSV25, BBK+25, XLLF25], Traveling Salesperson [PCT92], Chinese Postman[4] [HBvdLT99], Network Flow [HN16], and Set Cover [NYQ+25]. We focus on prior work most relevant to our analysis. Edelman et al. [EdLW24] showed that for monotonic TU games—games that satisfy $\nu(S) \leq \nu(T)$ if $S \subseteq T$—the core corresponds to the solution of a market

[4]It is similar to the aforementioned traveling salesman problem, except the agents own edges rather than vertices, and the goal is to traverse all edges in a coalition.

Table 3.1: Overview of the (parameterized) complexity of Core-Stability Verification and Core-Stability Existence for four Partitioned Combinatorial Optimization Games. Results in green are tractable, results in red are intractable ("-c" = complete, "-h" = hard). "P*" means that the corresponding game always admits a core-stable allocation, and finding one can be done in polynomial time. Results obtained in prior work are cited in the same cell; all other results are obtained in this thesis. References [T x], [P x], and [O x] point to Theorem, Proposition, and Observation x of this thesis, respectively. "–" denotes a case with no separate result.

Opt. problem Π	PCOG-Π-CV		PCOG-Π-CE	
MinVC	DP-c	[T3.16]	Θ_2^{P}-c	[T3.21, T3.23]
#agents $\hat{n}$	DP-c	[T3.16]	Θ_2^{P}-c	[T3.21, T3.23]
max ag-size $c_{\max}$	DP-c	[T3.16]	Θ_2^{P}-c	[T3.21, T3.24]
MinDS	DP-c	[T3.34]	Θ_2^{P}-c	[T3.39, T3.40]
#agents $\hat{n}$	DP-c	[T3.34]	Θ_2^{P}-c	[T3.39, T3.40]
max ag-size $c_{\max}$	DP-c	[T3.34]	Θ_2^{P}-c	[T3.39, T3.42]
MinST	coNP-c	[FKFH97], [O3.48]	P	[Bir76, GH81], [P3.12]
#agents $\hat{n}$	FPT	[P3.50]	–	–
max ag-size $c_{\max}$	coNP-c	[FKFH97], [O3.48]	–	–
MaxMatch	coNP-c	[BKPP19]	coNP-c	[BKPW18, BKPP19], [P3.49]
#agents $\hat{n}$	FPT	[P3.50]	FPT	[P3.51]
max ag-size $c_{\max}$	coNP-c	[BKPP19]	coNP-c	[BKPW18, BKPP19], [P3.49]

demand problem in which the goods are the coalitions. The core is then exactly the set of prices under which every agent prefers the good corresponding to the grand coalition.

Deng et al. [DIN99] study three relevant problems: CV, CE, and the search variant of CE in the context of COG. They establish a foundational characterization result for COGs: a core-stable allocation exists if and only if the underlying linear program has an integral optimal solution. For COG-MinVC, COG-MaxMatch, and COG-Independent Set, they provide polynomial-time algorithms to determine core-stability existence and find core allocations when possible. However, these polynomial-time results do not transfer to the generalized PCOG setting we study. For the vertex cover game, Fang and Kim [FKZ08] analyze several structural properties. They show that a core-stable allocation exists if and only if every edge belongs to a maximum matching, and reconfirm that CE is polynomial-time solvable for COG-Vertex Cover.

Van Velzen [vV04] introduces several variants of the dominating set game based on the domination distance k (the length of the shortest path from a dominating set vertex). The standard variant, where $k = 1$, corresponds to the restricted case of our COG-MinDS where each agent owns exactly one vertex. We confirm van Velzen's conjecture [vV04]

that determining core-stability existence in his setting is NP-hard (see Theorem 3.45).

For spanning tree games, Claus and Kleitman [CK73] introduce the original model, while Bird [Bir76] proves that a core-stable allocation always exists. Granot and Huberman [GH81] further show that such allocations can be computed in polynomial time from any minimum-cost spanning tree. However, Faigle et al. [FKFH97] establish that verifying core membership remains coNP-complete despite these positive results.

Most directly related to our work, Biró et al. [BKPP19, BBK$^+$25] extend COG-MaxMatch to the partitioned setting (PCOG) to model international kidney exchange. They establish that PCOG-MaxMatch is equivalent to COG-Max$b$Match for core stability analysis. Building on results from Biró et al. [BKPW18], they show that CV is coNP-complete and CE is coNP-hard. We complement the latter by proving coNP-containment for CE. Gourvès et al. [GMP12] study PCOG-Max$b$Match for another fairness concept, individual rationality (i.e., no agent prefers to be alone), and show that determining whether an individually rational allocation exists is NP-complete, even for bipartite graphs. Gerstbrein et al. [GSV25] study the problem of removing vertices to turn a given instance of edge-weighted COG-Max$b$Match into a yes-instance. Xiao et al. [XLLF25] study COG-Max$b$Match from an approximation point of view. They show that the best constant factor for the approximate core is characterized by the integrality gap of the underlying LP relaxation, and provide an LP-based algorithm to compute such approximate core-stable allocations.

The multi-located player model represents another approach to generalizing COGs, in which agents may own multiple elements, but the optimization goal considers only one element per agent. This model has been studied for the Chinese postman problem [EFH20], fixed tree games [MvVHN06], and Minimum Cost Spanning Tree [LNB16].

PCOG concepts have been applied to model multi-organization collaboration in scheduling [Rza07, PRT09, CCTW11, CCR14, DP21]. For more details, see Chapter 4.

Chapter outline. In Section 3.2, we introduce cooperative game theory concepts, define our four partitioned combinatorial optimization games, and present key structural properties that connect core stability to the underlying optimization problems. In Sections 3.3 to 3.5, we present our findings for the four games. In each section, we follow a similar structure: We first address CV and then CE. We conclude in Section 3.6 with preliminary results on individual rationality and directions for future research.

3.2 Preliminaries

Remark 3.1. *In this chapter, we use the term agents to refer to players in the games we study.*

Here, we present some additional definitions for this chapter. For background on cooperative games with transferable utility, see Section 2.4.2. In this chapter, we mainly

consider minimization games as defined in Definition 2.50, where each coalition has a cost rather than a utility. To highlight that the coalition value is a *cost* that the agents have to pay, we use c instead of ν for these games; the notions of preimputation, individual rationality, and the core are then exactly those of Definition 2.50 for minimization games. The maximization games we consider (see Section 3.5) use ν and the standard definitions without modification. As a reminder in the tuple (N, ν), N is the set of agents and ν is the characteristic function mapping every subset of agents to a non-negative real number.

We study the following two computational problems related to core stability:

Core-Stability Verification (CV)
Input: A TU-game (N, ν) and an allocation $\boldsymbol{\alpha} \in \mathbb{R}^{\hat{n}}$.
Question: Is $\boldsymbol{\alpha}$ core-stable?

Core-Stability Existence (CE)
Input: A TU-game (N, ν).
Question: Does (N, ν) admit a core-stable allocation?

Remark 3.2. *When the characteristic function ν is given explicitly—that is, a value (or cost) for every coalition—the input length is exponential in $\hat{n}$. Under this unrestricted encoding, CV is solvable in polynomial time, whereas CE lies in* NP*. Our focus, however, is on combinatorial optimization games whose value functions admit compact encodings. For such games, the input size is only polynomial in $\hat{n}$, making the two problems far more subtle and worthy of detailed complexity analysis.*

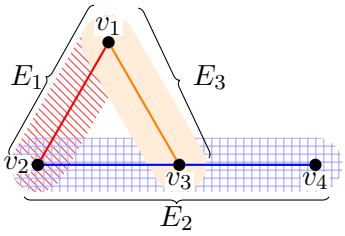
3.2.1 (Partitioned) Combinatorial Optimization Games

We consider Partitioned Combinatorial Optimization Games (PCOGs) that are defined on graphs. We refer to Section 2.1 for background on graphs. Combinatorial Optimization Games (COGs) are a class of games with compact representation of the value function, where we are provided with an undirected graph, and depending on the optimization problem, each agent owns either a vertex or an edge of the graph. The value of a subset of agents is based on the objective function of the underlying optimization problem, such as vertex cover size, on the subgraph induced by the agents' vertices or edges. *PCOG* extends COG by allowing agents to own a subset of vertices or edges instead of a single one. For the sake of brevity, we define the four PCOGs separately.

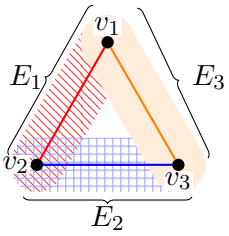
Definition 3.3 (Partitioned Minimum Vertex Cover Game)**.** *A Partitioned Minimum Vertex Cover Game (PCOG-MinVC) is a minimization game which consists of a set N of agents and an edge-partitioned graph $G = (V, E = E_1 \cup \cdots \cup E_{\hat{n}})$ such that every agent $i \in \mathsf{N}$ owns the edge set E_i. For every $S \subseteq \mathsf{N}$, the cost of S is defined as the size of a minimum vertex cover of $G[\bigcup_{i \in S} E_i]$; recall that a subset $V' \subseteq V$ of vertices is a vertex cover if each edge $e \in E$ is incident to at least one vertex in V', in which case we say that e is covered by V'.*

Example 3.4 (Partitioned Minimum Vertex Cover Game)**.** *Below are two PCOG-MinVCs on three agents* $1, 2,$ *and* 3 *with edge sets* $E_1, E_2,$ *and* E_3*, respectively.*

G_1:



G_2:



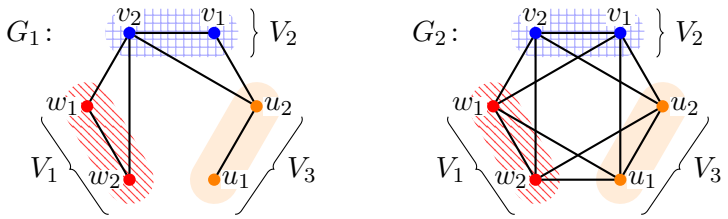
In G_1*, each agent owns edges whose minimum vertex cover has size* 1*, so* $c(\{i\}) = 1$ *for every* $i \in [3]$*. For the two coalitions* $\{1, 3\}$ *and* $\{2, 3\}$*, their edges can each be covered by a single shared vertex, so* $c(\{1, 3\}) = c(\{2, 3\}) = 1$*. To cover the edges in* $E_1 \cup E_2$ *two vertices are needed, and therefore* $c(\{1, 2\}) = 2$*. We need two vertices to cover the whole graph, and therefore* $c(\mathsf{N}) = 2$*. In order to be a preimputation, it must hold that* $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 = 2$*. Combining this with the constraint* $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_3 \leq 1$*, leads to* $\boldsymbol{\alpha}_2 \geq 1$*, and combining it with* $\boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 \leq 1$ *leads to* $\boldsymbol{\alpha}_1 \geq 1$*; since each agent can cover their edges using a single vertex, it holds that* $c(\{i\}) = 1$*, and we obtain* $\boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_2 = 1$ *and* $\boldsymbol{\alpha}_3 = 0$*. Hence* $\boldsymbol{\alpha} = (1, 1, 0)$ *is the unique core-stable allocation.*

The game G_2 *is a triangle in which each agent owns a single edge, and therefore* $c(\{i\}) = 1$*. Since every pair of edges in a triangle can be covered by a single vertex, it must hold that* $c(\{i, j\}) = 1$ *for every pair of agents* $i, j \in [3]$*. As a minimum vertex cover of the whole graph consists of two vertices*[5]*, it must hold that* $c(\mathsf{N}) = 2$*. A core-stable allocation would have to satisfy* $\boldsymbol{\alpha}_i + \boldsymbol{\alpha}_j \leq 1$ *for all three pairs. Summing up these inequalities leads to the constraint* $2(\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3) \leq 3$*, which implies* $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 \leq 1.5$*. This contradicts the preimputation condition that requires* $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 = 2$*. Therefore, no core-stable allocation exists, and the core is empty.*

Definition 3.5 (Partitioned Minimum Dominating Set Game)**.** *A* Partitioned Minimum Dominating Set Game (PCOG-MinDS) *is a minimization game which consists of a set* N *of agents and a vertex-partitioned graph* $G = (V = V_1 \cup \cdots \cup V_{\hat{n}}, E)$ *such that every agent* $i \in \mathsf{N}$ *owns the vertex set* V_i*. For every* $S \subseteq \mathsf{N}$*, the cost of* S *is defined as the size of a minimum dominating set of* $G[\bigcup_{i \in S} V_i]$*; recall that a vertex subset* $V' \subseteq V$ *is a dominating set if each vertex* $v \in V$ *is in* V' *or adjacent to at least one vertex in* V'*; in this case we say that* v *is* dominated by V'*.*

Example 3.6. *Below are two PCOG-MinDSs on three agents* $1, 2,$ *and* 3 *with vertex subsets* $V_1, V_2,$ *and* V_3*, respectively.*

[5] In this example, any two distinct vertices form a minimum vertex cover.



A minimum dominating set of G_1 has size 2 (e.g., $\{u_2, v_2\}$), so $c(\mathsf{N}) = 2$, and each agent's own two vertices form a single edge, so $c(\{i\}) = 1$ for every $i \in [3]$. The two coalitions $\{1,2\}$ and $\{2,3\}$ satisfy $c(\{1,2\}) = c(\{2,3\}) = 1$: In $G[V_1 \cup V_2]$ the vertex v_2 dominates all four vertices, and in $G[V_2 \cup V_3]$ the vertex u_2 does. The coalition $\{1,3\}$ requires two vertices to cover their subgraph, and therefore $c(\{1,3\}) = 2$.

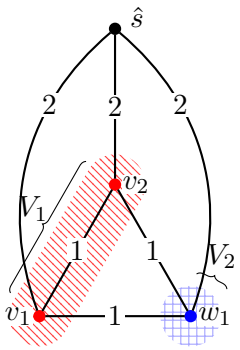
Core stability requires $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 = 2$ together with $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 \leq 1$ and $\boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 \leq 1$. The first inequality together with the preimputation condition implies $\boldsymbol{\alpha}_3 \geq 1$ and the second implies $\boldsymbol{\alpha}_1 \geq 1$. As $c(\{i\}) = 1$ for every $i \in [3]$, we get $\boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_3 = 1$ and $\boldsymbol{\alpha}_2 = 0$. Hence $\boldsymbol{\alpha} = (1,0,1)$ is the unique core-stable allocation.

Instance G_2 adds edges so that every pair of agents can be dominated by a single vertex, giving $c(\{i,j\}) = 1$ for all $i \neq j$, while the whole graph still requires two vertices to dominate, $c(\mathsf{N}) = 2$. As in G_2 of Example 3.4, summing the three pair inequalities $\boldsymbol{\alpha}_i + \boldsymbol{\alpha}_j \leq 1$ implies $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 \leq 1.5$, contradicting the preimputation condition $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 = 2$. Therefore, the core is empty.

Definition 3.7 (Partitioned Minimum Spanning Tree Game)**.** *A* Partitioned Minimum Spanning Tree Game (PCOG-MinST) *is a minimization game which consists of a set N of agents and a* complete *vertex-partitioned graph $G = (V = \{\hat{s}\} \cup V_1 \cup \cdots \cup V_{\hat{n}}, E)$, and a weight function $\omega \colon E \to \mathbb{R}_{\geq 0}$ such that every agent $i \in \mathsf{N}$ owns the vertex set V_i, but no one owns the supply node $\hat{s}$. For every $S \subseteq \mathsf{N}$, the cost of S is defined as the total weight of a minimum-weight spanning tree of $G[\{\hat{s}\} \cup \bigcup_{i \in S} V_i]$; recall that a subgraph T of G is a spanning tree if $V(T) = V(G)$ and T is a tree. The* weight *of T is defined as $\omega(T) \coloneqq \sum_{e \in E(T)} \omega(e)$.*

It is known that a core-stable allocation for COG-MinST (i.e., the unpartitioned variant of PCOG-MinST) always exists and can be found in polynomial time [Bir76, GH81]. We will see in Proposition 3.12 that the same is true for the partitioned variant PCOG-MinST.

Example 3.8. *Consider the following instance, with two agents and three vertices, as seen in the graph on the right.*

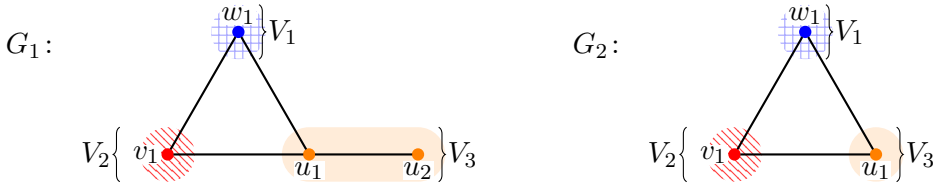


Connecting $\hat{s}$ to agent 2's single vertex w_1 costs 2, so $c(\{2\}) = 2$. Agent 1 owns v_1 and v_2; spanning $\{\hat{s}, v_1, v_2\}$ uses the edge $\{v_1, v_2\}$ of weight 1 and one weight-2 edge (either $\{v_1, \hat{s}\}$ or $\{v_2, \hat{s}\}$), so $c(\{1\}) = 3$. The minimum-weight spanning tree of the whole uses the cheap weight-1 edges between the agents' vertices and connects to $\hat{s}$ with a single weight-2 edge, and therefore $c(\mathsf{N}) = 4$.

Core stability requires $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 = 4$ and $\boldsymbol{\alpha}_1 \leq 3$ and $\boldsymbol{\alpha}_2 \leq 2$. This implies $\boldsymbol{\alpha}_1 \geq 2$, and $\boldsymbol{\alpha}_2 \geq 1$. Therefore it must hold that $\boldsymbol{\alpha}_1 = 2+\varepsilon$ and $\boldsymbol{\alpha}_2 = 2-\varepsilon$ with $\varepsilon \in [0, 1]$. Every such allocation charges each agent at most its standalone cost, so neither agent can improve by deviating, and the one unit of cost saved by cooperating—the gap between 5 and 4—is modeled by ε; the value that the agents are free to divide. Therefore, these allocations are exactly the core-stable allocations.

Definition 3.9 (Partitioned Maximum Matching Game)**.** *A Partitioned Maximum Matching Game (PCOG-MaxMatch) is a maximization game which consists of a set N of agents and a vertex-partitioned graph $G = (V = V_1 \cup \cdots \cup V_{\hat{n}}, E)$ such that every agent $i \in \mathsf{N}$ owns the vertex set V_i. For every $S \subseteq \mathsf{N}$, the value of S is defined as the size of a maximum matching of $G[\bigcup_{i \in S} V_i]$; recall that an edge subset $E' \subseteq E$ is a matching if all edges in E' are pairwise disjoint.*

Example 3.10. *For PCOG-MaxMatch, we will use the same graphs from Example 3.4, but divide the vertices rather than the edges:*



Note that, unlike the other games, this is a maximization game (cf. Definition 2.50). I.e., the agents prefer larger values.

In G_1 the graph has a maximum matching of size 2, so $\nu(\mathsf{N}) = 2$. Agent 3 can match its two vertices, and therefore $\nu(\{3\}) = 1$. Agents 1 and 2 each own a single vertex and so have value 0 in isolation, i.e., $\nu(\{1\}) = \nu(\{2\}) = 0$. It holds that $\nu(\{i, j\}) = 1$ for all $i \neq j \in [3]$, i.e., each coalition of size two can obtain a utility of 1. Core stability requires that $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 = 2$, $\boldsymbol{\alpha}_3 \geq 1$ and $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 \geq 1$. Every allocation that satisfies these three constraints is in the core. The allocation $\boldsymbol{\alpha} = (0.5, 0.5, 1)$ is one such example.

Instance G_2 is a triangle, with the vertices partitioned one per agent. Every pair of agents can match their vertices, so $\nu(\{i, j\}) = 1$, but a triangle admits a matching of only one edge, so $\nu(\mathsf{N}) = 1$. Summing the three coalition inequalities $\boldsymbol{\alpha}_i + \boldsymbol{\alpha}_j \geq 1$ implies $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 \geq 1.5$, contradicting the preimputation condition $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 = 1$. Therefore, the core is empty.

Note that the decision variant of whether a given graph has a vertex cover or a dominating set of size at most k, referred to as Dominating Set and Vertex Cover, is NP-complete [GJ79]. In contrast, both a minimum-weight spanning tree [Pri57] and a maximum-size matching can be found in polynomial time [Edm65].

We use COG-MinVC, COG-MinDS, COG-MinST, and COG-MaxMatch to refer to the unpartitioned classes of PCOG-MinVC, PCOG-MinDS, PCOG-MinST, and PCOG-MaxMatch respectively, i.e., when each agent owns a single unit of the graph.

Given a game class X, we define problems X-CE and X-CV by restricting the input game to the game class X. For example, the problem of deciding whether an instance of PCOG-MinVC admits a core-stable allocation is referred to as PCOG-MinVC-CE.

Parameters. For the parameterized complexity study, we focus on two parameters: (1) the number of agents: $\hat{n} \coloneqq |\mathsf{N}|$ and (2) the maximum agent size c_{max}, which is $c_{max} \coloneqq \max_{i \in \mathsf{N}} |V_i|$ for PCOG-MinDS, PCOG-MinST, and PCOG-MaxMatch, and $c_{max} \coloneqq \max_{i \in \mathsf{N}} |E_i|$ for PCOG-MinVC.

3.2.2 Structural Observations

In this section, we present a few structural observations for our PCOGs. First, we prove an auxiliary condition for when a game has a non-empty core, with the help of another game.

Proposition 3.11. *Let $G_1 = (X, \nu_1)$ and $G_2 = (Y, \nu_2)$ be two maximization games or two minimization games, $b \geq 0$ a non-negative constant, and $f \colon X \to Y$ a surjective function such that $\nu_2(S) = b \cdot \nu_1(f^{-1}(S))$ holds for all coalitions $S \subseteq Y$. If $\boldsymbol{\alpha} \colon X \to \mathbb{R}_{\geq 0}$ is a core-stable allocation for G_1, then in $(\max(|X|, |Y|))^{O(1)}$ time we can derive a core-stable allocation for G_2 from $\boldsymbol{\alpha}$.*

Proof. Assume that the allocation $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{|X|})$ is core-stable for G_1. We define an allocation $\boldsymbol{\alpha}'$ for G_2 and show that it is core-stable for G_2. For each agent $j \in Y$, we set $\boldsymbol{\alpha}'_j \coloneqq b \cdot \sum_{i \in f^{-1}(j)} \boldsymbol{\alpha}_i$. First, we note that since f is surjective, $\sum_{j \in S} \boldsymbol{\alpha}'_j = b \cdot \sum_{j \in S} \sum_{i \in f^{-1}(j)} \boldsymbol{\alpha}_i = b \cdot \sum_{i \in f^{-1}(S)} \boldsymbol{\alpha}_i$ holds for all coalitions $S \subseteq Y$.

Due to this, $\boldsymbol{\alpha}'$ is a preimputation for G_2 because $\sum_{j \in Y} \boldsymbol{\alpha}'_j = b \cdot \sum_{i \in f^{-1}(Y)} \boldsymbol{\alpha}_i = b \cdot \sum_{i \in X} \boldsymbol{\alpha}_i = b \cdot \nu_1(X) = \nu_2(Y)$; recall that the second equation holds since f is surjective. It remains to show that there is no blocking coalition. Consider an arbitrary coalition $S \subset Y$. If G_1 and G_2 are both maximization games, it holds that $\nu_2(S) = b \cdot \nu_1(f^{-1}(S)) \leq b \cdot \sum_{i \in f^{-1}(S)} \boldsymbol{\alpha}_i = \sum_{j \in S} \boldsymbol{\alpha}'_j$; the inequality $b \cdot \nu_1(f^{-1}(S)) \leq b \cdot \sum_{i \in f^{-1}(S)} \boldsymbol{\alpha}_i$ holds since $\boldsymbol{\alpha}$ is core-stable, implying that S is not blocking $\boldsymbol{\alpha}'$, as desired. Similarly, if G_1 and G_2 are both minimization games, it holds that $\nu_2(S) = b \cdot \nu_1(f^{-1}(S)) \geq b \cdot \sum_{i \in f^{-1}(S)} \boldsymbol{\alpha}_i = \sum_{j \in S} \boldsymbol{\alpha}'_j$; the inequality $b \cdot \nu_1(f^{-1}(S)) \geq b \cdot \sum_{i \in f^{-1}(S)} \boldsymbol{\alpha}_i$ holds since $\boldsymbol{\alpha}$ is core-stable, implying that S is not blocking $\boldsymbol{\alpha}'$, as desired. □

Proposition 3.11, together with the fact that every instance of COG-MinST admits a core-stable allocation which can be found in polynomial time [Bir76, GH81], implies the following:

Proposition 3.12. *For every $\mu \in \{\textsc{MinVC}, \textsc{MinDS}, \textsc{MinST}, \textsc{MaxMatch}\}$, if an instance of COG-$\mu$ consisting of an unpartitioned graph G admits a core-stable allocation, then so does any instance of PCOG-μ which consists of a graph that is an edge- or vertex-partition of G. In particular, every instance of PCOG-MinST admits a core-stable allocation, which can be found in polynomial time.*

Proof. We show the proof for $\mu \in \{\textsc{MinDS}, \textsc{MinST}, \textsc{MaxMatch}\}$; the proof for $\mu = \textsc{MinVC}$ works in the same way by mapping the edges to the partition rather than the vertices. Let $G = (V, E)$ be the original unpartitioned graph, and let $\hat{G} = (V = V_1 \cup \cdots \cup V_{\hat{n}}, E)$ be a vertex-partition of G. Let $b = 1$. We define a surjective function $f \colon V \to [\hat{n}]$ in the following way: For every $v \in V$, we set $f(v) = \ell$, where ℓ is the agent such that $v \in V_\ell$; observe that $f^{-1}(\ell) = V_\ell$. As $V_1, \ldots, V_{\hat{n}}$ is a partition of V, the function f is surjective. If $\mu = \textsc{MinDS}$, then for every coalition $S \subseteq [\hat{n}]$, it holds that $\bigcup_{i \in S} V_i = f^{-1}(S)$, and thus the size of a minimum dominating set of $\hat{G}[\bigcup_{i \in S} V_i]$ is equal to the size of a minimum dominating set of $G[f^{-1}(S)]$, as required. Similarly, if $\mu \in \{\textsc{MinST}, \textsc{MaxMatch}\}$, for every coalition $S \subseteq [\hat{n}]$, it holds that $\bigcup_{i \in S} V_i = f^{-1}(S)$, and thus the size of a minimum spanning tree (resp. maximum matching) of $\hat{G}[\bigcup_{i \in S} V_i]$ is equal to the size of a minimum spanning tree (resp. maximum matching) of $G[f^{-1}(S)]$, as required.

Therefore, we have found a surjective function f that satisfies $\nu_2(S) = \nu_1(f^{-1}(S))$, and per Proposition 3.11, shown the first statement.

Since every instance of COG-MinST admits a core-stable allocation [Bir76, GH81], the first statement yields that every instance of PCOG-MinST admits a core-stable allocation, which can moreover be found in polynomial time. □

Finally, we show that if the agents of an instance of a partitioned game can be partitioned into two subsets such that the subgraphs induced by the vertices or edges of these agents are not pairwise connected, then we can treat the instance restricted to these agent sets as two separate instances. This result will be useful for our hardness proofs.

Observation 3.13. *Let $(\mathsf{N}, G = (V, E))$ be an instance of PCOG-MinVC, PCOG-MinDS, PCOG-MinST, or PCOG-MaxMatch such that G is the disjoint union of two graphs G_1 and G_2. Let N_1 and N_2 be the sets of agents owning at least one vertex (or edge) in G_1 and G_2, respectively. If $\mathsf{N}_1 \cap \mathsf{N}_2 = \emptyset$, then an allocation $\boldsymbol{\alpha}$ is core-stable if and only if $\boldsymbol{\alpha}$ restricted to N_1 and $\boldsymbol{\alpha}$ restricted to N_2 are both core-stable.*

Proof. We show the proof for maximization games; the proof for minimization games is analogous by flipping the inequalities. First, assume that $\boldsymbol{\alpha}$ is core-stable on N. Since $\boldsymbol{\alpha}$ is a preimputation, we have that $\nu(\mathsf{N}) = \sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i$. Because there are no edges

between G_1 and G_2, we also have that $\nu(\mathsf{N}_1) + \nu(\mathsf{N}_2) = \nu(\mathsf{N})$. If $\nu(\mathsf{N}_1) < \sum_{i \in \mathsf{N}_1} \boldsymbol{\alpha}_i$, then $\nu(\mathsf{N}_2) > \sum_{i \in \mathsf{N}_2} \boldsymbol{\alpha}_i$ and N_2 blocks $\boldsymbol{\alpha}$. If $\nu(\mathsf{N}_1) > \sum_{i \in \mathsf{N}_1} \boldsymbol{\alpha}_i$, then N_1 blocks $\boldsymbol{\alpha}$. Thus $\boldsymbol{\alpha}$ restricted on N_1 must be a preimputation. By the same identity $\nu(\mathsf{N}_1) + \nu(\mathsf{N}_2) = \nu(\mathsf{N})$, if $\nu(\mathsf{N}_2) \neq \sum_{i \in \mathsf{N}_2} \boldsymbol{\alpha}_i$ then N_1 or N_2 blocks; hence $\boldsymbol{\alpha}$ restricted to N_2 is a preimputation. Moreover, if a coalition $S \subseteq \mathsf{N}_1$ or $S \subseteq \mathsf{N}_2$ blocks $\boldsymbol{\alpha}$ restricted to N_1 or N_2, respectively, then clearly S also blocks $\boldsymbol{\alpha}$. Thus $\boldsymbol{\alpha}$ restricted to N_1 and $\boldsymbol{\alpha}$ restricted to N_2 must both be core-stable.

Now assume that $\boldsymbol{\alpha}$ restricted to N_1 is core-stable, and $\boldsymbol{\alpha}$ restricted to N_2 as well. We obtain that $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = \sum_{i \in \mathsf{N}_1} \boldsymbol{\alpha}_i + \sum_{i \in \mathsf{N}_2} \boldsymbol{\alpha}_i = \nu(\mathsf{N}_1) + \nu(\mathsf{N}_2) = \nu(\mathsf{N})$, and thus $\boldsymbol{\alpha}$ is a preimputation. Moreover, if a coalition S blocks $\boldsymbol{\alpha}$, we have that $\sum_{i \in S \cap \mathsf{N}_1} \boldsymbol{\alpha}_i + \sum_{i \in S \cap \mathsf{N}_2} \boldsymbol{\alpha}_i = \sum_{i \in S} \boldsymbol{\alpha}_i < \nu(S) = \nu(S \cap \mathsf{N}_1) + \nu(S \cap \mathsf{N}_2)$, and thus we must have that $\sum_{i \in S \cap \mathsf{N}_1} \boldsymbol{\alpha}_i < \nu(S \cap \mathsf{N}_1)$ or $\sum_{i \in S \cap \mathsf{N}_2} \boldsymbol{\alpha}_i < \nu(S \cap \mathsf{N}_2)$, implying that $\boldsymbol{\alpha}$ restricted to N_1 or N_2 is not core-stable, a contradiction. □

We conclude this section with a result regarding super-additive games and individually rational allocation. For a definition of individual rationality, see Definition 2.51.

Definition 3.14 (Super-additive games)**.** *We say that a TU-game (N, ν) is super-additive if for every pair of subsets of agents $X, Y \subseteq \mathsf{N}$ such that $X \cap Y = \emptyset$, it holds that (i) if the game is a maximization game, then $\nu(X \cup Y) \geq \nu(X) + \nu(Y)$, (ii) if the game is a minimization game, then $c(X \cup Y) \leq c(X) + c(Y)$.*

Observation 3.15. *Every super-additive game admits an individually rational allocation.*

Proof. We first show the proof for maximization games. Let $\mathsf{N} = [\hat{n}]$ be the set of agents. Since the game is super-additive, it holds that $\sum_{i \in \mathsf{N}} \nu(\{i\}) \leq \nu(\mathsf{N})$. Let $\beta = \frac{\nu(\mathsf{N}) - \sum_{i \in \mathsf{N}} \nu(\{i\})}{\hat{n}}$; note that $\beta \geq 0$. Then the allocation $\boldsymbol{\alpha} = (\nu(\{1\}) + \beta, \ldots, \nu(\{\hat{n}\}) + \beta)$ is a preimputation and individually rational, as for each agent $i \in \mathsf{N}$ it holds that $\boldsymbol{\alpha}_i \geq \nu(\{i\})$.

The proof for the minimization games is similar, except we need to ensure that no agent receives a negative cost. Let $\mathsf{N} = [\hat{n}]$ be the set of agents. We assume that $c(\mathsf{N}) > 0$, since if $c(\mathsf{N}) = 0$, we can assign cost 0 to every agent and receive an individually rational allocation. Let $r = \frac{c(\mathsf{N})}{\sum_{i \in \mathsf{N}} c(\{i\})}$. Since the game is super-additive, it holds that $\sum_{i \in \mathsf{N}} c(\{i\}) \geq c(\mathsf{N})$ and thus $0 < r \leq 1$. Consider the allocation $\boldsymbol{\alpha} = (c(\{1\}) \cdot r, \ldots, c(\{\hat{n}\}) \cdot r)$. This is a preimputation: $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = \sum_{i \in \mathsf{N}} c(\{i\}) \cdot r = r \sum_{i \in \mathsf{N}} c(\{i\}) = \frac{c(\mathsf{N})}{\sum_{i \in \mathsf{N}} c(\{i\})} \sum_{i \in \mathsf{N}} c(\{i\}) = c(\mathsf{N})$. Since $0 < r \leq 1$, for every agent $i \in \mathsf{N}$ it holds that $0 \leq \boldsymbol{\alpha}_i \leq c(\{i\})$, and $\boldsymbol{\alpha}$ is individually rational. □

PCOG-MinVC, PCOG-MinDS, PCOG-MinST, and PCOG-MaxMatch are all super-additive, since the union of solutions for any two subsets of vertices (resp. edges) is a solution for the union of the two subsets. This directly implies that these games always admit an individually rational allocation.

3.3 Partitioned Minimum Vertex Cover Game

In this section, we consider partitioned minimum vertex cover games. We start with CV and show that PCOG-MinVC-CV is DP-complete. This holds even if the number of agents or the agent size is a constant. We then turn to CE. We show that PCOG-MinVC-CE is Θ_2^{P}-complete. The hardness again holds even if the number of agents or the agent size is constant.

3.3.1 Core-Stability Verification

To verify whether a candidate allocation is core-stable, we need to solve an NP problem—determining whether a vertex cover of that size exists in the graph—and then address a complementary coNP problem, certifying that no induced subgraph admits a strictly smaller vertex cover. Because these two conditions must hold simultaneously—or in other words, the problem is the intersection of an NP- and coNP-problem—PCOG-MinVC-CV belongs to the class DP and is, in fact, DP-complete.

Theorem 3.16. *PCOG-MinVC-CV is* DP*-complete. The problem remains* DP*-hard even if* $\hat{n} = 1$ *or* $c_{max} = 7$.

Proof. The proof proceeds in three parts. First, we show DP-membership, and afterward, we present two reductions, the first utilizing only one agent, but with an unbounded number of vertices, and the second only utilizing agents with at most 7 vertices, but an unbounded number of them.

DP**-membership.** We first show DP-membership. By Papadimitriou [Pap94], we need to provide two NP-problems P and Q, and show that PCOG-MinVC-CV $= P \cap \overline{Q}$, in other words, every instance of PCOG-MinVC-CV is a yes-instance if and only if it is a *yes*-instance of P but a *no*-instance of Q. To this end, let P and Q be the following two NP-problems.

Problem P**:** Given an instance $I = (G = (V, E_1 \cup \cdots \cup E_{\hat{n}}), \boldsymbol{\alpha})$, does G admit a vertex cover of size at most $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i$?

Problem Q**:** Given an instance $I = (G = (V, E_1 \cup \cdots \cup E_{\hat{n}}), \boldsymbol{\alpha})$, is there a subset $S \subseteq \mathsf{N}$ of agents such that the induced graph $G[\cup_{i \in S} E_i]$ has a vertex cover with size strictly smaller than $\sum_{i \in S} \boldsymbol{\alpha}_i$?

It is straightforward to check that both problems are contained in NP since the witness of either problem has polynomial size and can be verified in polynomial time as well.

It remains to show that for each instance $I = (G = (V, E_1 \cup \cdots \cup E_{\hat{n}}), \boldsymbol{\alpha})$ of PCOG-MinVC-CV $\boldsymbol{\alpha}$ is core-stable if and only if I is a yes-instance of P and a no-instance of Q. By definition, allocation $\boldsymbol{\alpha}$ is core-stable if and only if $\boldsymbol{\alpha}$ is a preimputation and no subset of agents is blocking $\boldsymbol{\alpha}$. The latter is equivalent to I being a yes-instance of P

and no subset of agents having a vertex cover with size that is strictly smaller than their cost under the allocation $\boldsymbol{\alpha}$ (i.e., instance I being a no-instance of Q), as desired.

DP-hardness when $\hat{n} = 1$. We now show DP-hardness for the restricted case where there is only one agent. We achieve this by providing a polynomial-time reduction from the SAT-UNSAT problem; see Section 2.7. Intuitively, we will encode the two formulas in a way that an unsatisfiable formula requires one more vertex for a minimum vertex cover than a satisfiable formula. We then set the allocation of the organization in a way that it is a preimputation if and only if we have a yes-instance of SAT-UNSAT.

Let (φ_1, φ_2) be an instance of SAT-UNSAT which are over the variable sets X_1 and X_2, respectively. We create an instance $I' = (G, \boldsymbol{\alpha})$ of PCOG-MinVC-CV with only one agent. The graph G consists of three disjoint subgraphs: G_1, $G_2^{(1)}$, and $G_2^{(2)}$, where G_1 corresponds to φ_1 and the remaining two $G_2^{(1)}$ and $G_2^{(2)}$ to φ_2. We note that each subgraph is a slight modification of the standard reduction from 3SAT to Vertex Cover.

Graph G_1. We create the following vertices for G_1:

- For every variable $x \in X_1$, create a positive literal vertex v_x and a negative literal vertex $v_{\overline{x}}$.
- For every clause $C \in \varphi_1$ create a dummy vertex $v_{C,d}$ and for every literal $\ell \in C$, create a vertex v_C^ℓ.
- Create a special vertex v^*.

We create the following edges for G_1:

- The two vertices corresponding to a variable are adjacent: For every $x \in X_1$, add the edge $\{v_x, v_{\overline{x}}\}$.
- For every clause $C \in \varphi_1$, the four vertices $\{v_C^\ell \mid \ell \in C\} \cup \{v_{C,d}\}$ form a clique.
- Each vertex corresponding to a literal in a clause is adjacent to the vertex corresponding to that literal: For every clause $C \in \varphi_1$, every literal $\ell \in C$, add edge $\{v_C^\ell, v_\ell\}$.
- The special vertex v^* is adjacent to all dummy vertices of clauses: For every clause $C \in \varphi_1$, add edge $\{v^*, v_{C,d}\}$.

Graphs $G_2^{(1)}$ and $G_2^{(2)}$. These two graphs are constructed in the same way as G_1, except that φ_2 and X_2 are the underlying CNF formula and variable set.

For the sake of brevity, we define the number of clauses in the formulas as $m_1 \coloneqq |\varphi_1|$ and $m_2 \coloneqq |\varphi_2|$, and the number of variables in the formulas $n_1 \coloneqq |X_1|$ and $n_2 \coloneqq |X_2|$.

Finally, we construct the cost vector $\boldsymbol{\alpha} = ((n_1 + 3m_1) + 2(n_2 + 3m_2) + 2)$; note that we have only one agent, so the vector has length one. Let I' denote the created instance $I' = (G_1 \uplus G_2^{(1)} \uplus G_2^{(2)}, \boldsymbol{\alpha})$. The following claim builds a connection between the satisfiability of a formula and the size of a minimum vertex cover.

Claim 3.17. *If φ_1 (resp. φ_2) is satisfiable, then every minimum vertex cover of G_1 (resp. $G_2^{(1)}$ and $G_2^{(2)}$ each) has n_1+3m_1 (resp. n_2+3m_2) vertices. If φ_1 (resp. φ_2) is unsatisfiable, then every minimum vertex cover of G_1 (resp. $G_2^{(1)}$ and $G_2^{(2)}$) has $n_1 + 3m_1 + 1$ (resp. $n_2 + 3m_2 + 1$) vertices.*

Proof. We show the statements only for G_1; the reasoning for $G_2^{(1)}$ and $G_2^{(2)}$ is analogous: The graphs are constructed using the same gadgets as G_1 but for φ_2. First, observe that for every variable $x \in X_1$, due to the edge between v_x and $v_{\overline{x}}$, one of the two corresponding vertices needs to be in the vertex cover. Since the vertices $\{v_C^\ell \mid \ell \in C\} \cup \{v_{C,d}\}$ form a clique, at least three of the vertices must be in any vertex cover. Therefore, every vertex cover of G_1 has to have size at least $n_1 + 3m_1$.

For the first statement, assume that φ_1 is satisfiable, and let $\sigma\colon X_1 \to \{\mathsf{T}, \mathsf{F}\}$ a satisfying assignment for φ_1. We construct a vertex cover V' of size $n_1 + 3m_1$ as follows: For each variable $x \in X_1$, if $\sigma(x) = \mathsf{T}$, then add v_x to V'; otherwise add $v_{\overline{x}}$ to V'. For each clause $C \in \varphi_1$, there must be at least one $\ell'_C \in C$ such that ℓ'_C is true under σ. We add the vertices in $\{v_C^\ell \mid \ell \in C \setminus \{\ell'_C\}\} \cup \{v_{C,d}\}$ to V'. We have that $|V'| = n_1 + 3m_1$, and thus if it is a vertex cover, it is a minimum vertex cover. It is also straightforward to verify that V' is a vertex cover: For each variable $x \in X_1$, the edge between them is covered. For each clause $C \in \varphi_1$, the edges between vertices corresponding to this clause are covered, because we have all but one of the corresponding vertices in V'. For each $C \in \varphi_1$, $\ell \in C$, the edge $\{v_\ell, v_C^\ell\}$ is covered by v_ℓ if ℓ is true under σ, otherwise it is covered by v_C^ℓ. Finally, each edge adjacent to v^* is also covered, because every dummy vertex is in V'.

Now we show the second statement. Graph G_1 has a vertex cover of size $n_1 + 3m_1 + 1$, as $V' = \{v_x \mid x \in X_1\} \cup \{v_C^\ell \mid C \in \varphi_1, \ell \in C\} \cup \{v^d\}$ forms a vertex cover of size n_1+3m_1+1: For every $x \in X_1$, the edge between v_x and $v_{\overline{x}}$ is covered by v_x. For every $C \in \varphi_1$, the edges between the vertices in $\{v_C^\ell \mid \ell \in C\} \cup \{v_{C,d}\}$ are covered by $\{v_C^\ell \mid \ell \in C\}$, the edge $\{v_{C,d}, v^*\}$ is covered by v^*, and for every $\ell \in C$, the edge $\{v_C^\ell, v_\ell\}$ is covered by v_C^ℓ.

Now, to show that this is indeed minimum, we show that if G_1 has a vertex cover of size $n_1 + 3m_1$, then the formula φ_1 is satisfiable. As we established previously, the set V' must contain for every $x \in X_1$ one of the vertices $v_x, v_{\overline{x}}$ and for every $C \in \varphi_1$ three of the vertices $\{v_C^\ell \mid \ell \in C\} \cup \{v_{C,d}\}$. The vertex v^* cannot be contained in V', as then the size would exceed $n_1 + 3m_1$. Let us construct an assignment $\sigma\colon X_1 \to \{\mathsf{T}, \mathsf{F}\}$ as follows: $\sigma(x_i) = \mathsf{T}$ if $v_{x_i} \in V'$ and $\sigma(x_i) = \mathsf{F}$ otherwise. We claim that σ is a satisfying assignment. Consider an arbitrary clause $C \in \varphi_1$. We know that exactly three of the vertices $\{v_C^\ell \mid \ell \in C\} \cup \{v_{C,d}\}$ must be in V'. Since $v^* \notin V'$, exactly one of $\{v_C^\ell \mid \ell \in C\}$ is not in V'. Let $v_C^{\ell'} \notin V'$. Since V' is a vertex cover, we must have that $v_{\ell'} \in V'$, and thus ℓ' is true under σ, satisfying the clause C. Thus σ is a satisfying assignment.

(end of the proof of Claim 3.17) ⋄

Using Claim 3.17, we can show the correctness, i.e., instance (φ_1, φ_2) is a yes-instance of SAT-UNSAT if and only if I' is a yes-instance of PCOG-MinVC-CV.

For the "only if" direction, assume that (φ_1, φ_2) is a yes-instance, i.e., φ_1 is satisfiable but φ_2 not. By Claim 3.17, graph G_1 has a minimum vertex cover of size $n_1 + 3m_1$ while $G_2^{(1)}$ (resp. $G_2^{(2)}$) has a minimum vertex cover of size $n_2 + 3m_2 + 1$. This implies that every minimum vertex cover for the entire graph has size $(n_1 + 3m_1) + 2(n_2 + 3m_2) + 2$. By definition, allocation $\boldsymbol{\alpha} = ((n_1 + 3m_1) + 2(n_2 + 3m_2) + 2)$ is a preimputation and also core-stable, witnessing that I' is a yes-instance.

For the "if" direction, assume (φ_1, φ_2) is a no-instance. Then φ_1 is not satisfiable, or φ_2 is satisfiable.

If φ_1 is not satisfiable, then by the second statement of Claim 3.17, the size of a minimum vertex cover for G_1 is $n_1 + 3m_1 + 1$. Again by Claim 3.17, every minimum vertex cover of the entire graph $G_1 \uplus G_2^{(1)} \uplus G_2^{(2)}$ has size either $n_1 + 3m_1 + 2(n_2 + 3m_2) + 3$ (if φ_2 is not satisfiable) or $n_1 + 3m_1 + 2(n_2 + 3m_2) + 1$ (if φ_2 is satisfiable). This implies that $\boldsymbol{\alpha}$ is not a preimputation, and hence not core-stable.

If φ_1 is satisfiable, then φ_2 must be satisfiable in order to be a no-instance. By Claim 3.17, every minimum vertex cover of the entire graph $G_1 \uplus G_2^{(1)} \uplus G_2^{(2)}$ has size $n_1 + 3m_1 + 2(n_2 + 3m_2)$, again implying that $\boldsymbol{\alpha}$ is not a preimputation, and hence not core-stable.

As we conclude in both cases that $\boldsymbol{\alpha}$ is not core-stable, instance I' is a no-instance as well.

DP-hardness when $c_{\max} = 7$. We now show DP-hardness for the restricted case in which each agent owns at most 7 edges. We assume that every literal appears at most twice, which we can ensure by applying the reduction from Berman et al. [BKS03] to φ_1 and φ_2. The reduction uses ideas similar to the previous proof, but we add a gadget that prevents a set of agents that does not contain all the variable agents from blocking.

Let (φ_1, φ_2) be an instance of SAT-UNSAT with variable sets X_1 and X_2, respectively. We create an instance $I' = (G = (V, E_1 \cup \cdots \cup E_{\hat{n}}), \boldsymbol{\alpha})$ of PCOG-MinVC-CV with $c_{\max} = 7$. The graph G consists of two disjoint subgraphs G_1 and G_2, where G_1 corresponds to φ_1 and G_2 to φ_2. We note that each subgraph is a slight modification of the standard reduction from 3SAT to Vertex Cover.

In what follows, let $n_1 \coloneqq |X_1|$, $n_2 \coloneqq |X_2|$, $m_1 \coloneqq |\varphi_1|$, and $m_2 \coloneqq |\varphi_2|$.

Graph G_1. We create the following vertices for G_1:

- For every variable $x_i \in X_1$, we create a positive literal vertex v_{x_i}, a negative literal vertex $v_{\overline{x}_i}$, and three chain vertices v_i^1, v_i^2, and v_i^3.

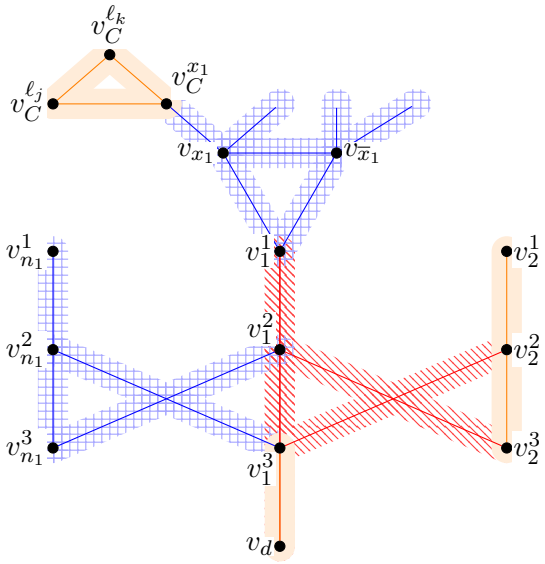


Figure 3.1: Illustration for the proof of Theorem 3.16 when $c_{\max} = 7$, showing the gadget for one variable x_1 occurring in the clause $C = (x_1 \vee \ell_j \vee \ell_k)$. The positive and negative literal vertices $v_{x_1}, v_{\overline{x}_1}$ and the clause vertices $v_C^{x_1}, v_C^{\ell_j}, v_C^{\ell_k}$ are joined through the chain vertices v_1^1, v_1^2, v_1^3; the patterns around the edges indicate which of the seven agents owns each edge set. Also visible are the chain vertices corresponding to x_2 and x_{n_1}.

- For every clause $C \in \varphi_1$ and every literal $\ell \in C$, we add the clause vertex v_C^{ℓ}.
- Additionally, we create a special vertex v^*.

We create the following edges for G_1:

- For every variable $x_i \in X_1$, let $\{v_{x_i}, v_{\overline{x}_i}, v_i^1\}$ form a clique, and add the edges $\{v_i^1, v_i^2\}, \{v_i^2, v_i^3\}, \{v_i^3, v_{i+1}^2\}, \{v_i^2, v_{i+1}^3\}$, where $n_1 + 1 = 1$.
- For every clause $C = (\ell_1 \vee \ell_2 \vee \ell_3) \in \varphi_1$, let $\{v_C^{\ell_1}, v_C^{\ell_2}, v_C^{\ell_3}\}$ form a clique. Also add the edges $\{\{v_{x_i}, v_C^{x_i}\} \mid C \in \varphi_1, x_i \in C\} \cup \{\{v_{\overline{x}_i}, v_C^{\overline{x}_i}\} \mid C \in \varphi_1, \overline{x}_i \in C\}$.
- Add the edge $\{v_d, v_1^3\}$.

Let us finally construct the agents:

- For every variable $x_i \in X_1$, add a variable agent a_{x_i}. This agent owns the edges in the clique formed by $\{v_{x_i}, v_{\neg x_i}, v_i^1\}$, and the edges $\{\{v_{x_i}, v_C^{x_i}\} \mid C \in \varphi_1, x_i \in C\} \cup \{\{v_{\overline{x}_i}, v_C^{\overline{x}_i}\} \mid C \in \varphi_1, \overline{x}_i \in C\}$, in total at most seven edges, as each literal appears in at most two clauses.

- For every variable $x_i \in X_1$, add a chain agent a_i^*, who owns the edges $\{v_i^1, v_i^2\}, \{v_i^2, v_i^3\}, \{v_i^3, v_{i+1}^2\}, \{v_i^2, v_{i+1}^3\}$, in total four edges.
- For every clause $C = (\ell_1 \vee \ell_2 \vee \ell_3) \in \varphi_1$, we add a clause agent a_C, who owns the edges in the clique formed by $\{v_C^{\ell_1}, v_C^{\ell_2}, v_C^{\ell_3}\}$, in total three edges.
- We add a dummy agent a_d, who owns the edge $\{v_d, v_1^3\}$.

Graph G_2. This graph is constructed in the same way as G_1, except that φ_2 and X_2 are the underlying CNF formula and variable set. Let N_1 be the set of agents constructed when constructing G_1 and N_2 the set of agents constructed when constructing G_2.

We construct the allocation $\boldsymbol{\alpha}$ as follows:

- For every $z \in [2]$:
  - In the graph G_z, for every $x_i \in X_z$, we allocate a_{x_i} cost 2 and allocate a_i^* cost 1.
  - In the graph G_z, for every $C \in \varphi_z$, we allocate a_C cost 2.
- In the graph G_1, we allocate a_d cost 0.
- In the graph G_2, we allocate a_d cost 1.

We first show the following two claims, which establish a connection between our construction and the SAT-formulas.

Claim 3.18. *If φ_1 (resp. φ_2) is satisfiable, then G_1 (resp. G_2) has a vertex cover of size $3n_1 + 2m_1$ (resp. $3n_2 + 2m_2$).*

Proof. We show the statement for G_1. The reasoning for G_2 is analogous: The graph is generated using the same gadgets as G_1 but over φ_2. We construct a vertex cover V' of G_1 of size $3n_1 + 2m_1$. Let $\sigma\colon X_1 \to \{\mathsf{T}, \mathsf{F}\}$ be a satisfying assignment of φ_1. For every variable $x_i \in X_1$, if σ assigns x_i to true, we add v_{x_i} to V', otherwise we add $v_{\overline{x}_i}$. Regardless, we add v_i^1 and v_i^3 to V'. For every clause $C = (\ell_1 \vee \ell_2 \vee \ell_3) \in \varphi_1$, there must be $w \in [3]$ such that ℓ_w is true under σ. We add $\{v_C^{\ell_1}, v_C^{\ell_2}, v_C^{\ell_3}\} \setminus \{v_C^{\ell_w}\}$ to V'. Clearly, $|V'| = 3n_1 + 2m_1$.

Clearly, for every $x_i \in X_1$, the edges in the clique formed by $v_{x_i}, v_{\overline{x}_i}$, and v_i^1 are covered. For every $C = (\ell_1 \vee \ell_2 \vee \ell_3) \in \varphi_1$, the edges in the clique formed by $\{v_C^{\ell_1}, v_C^{\ell_2}, v_C^{\ell_3}\}$ are also covered. For every $w \in [3]$, the edge $\{v_{\ell_w}, v_C^{\ell_w}\}$ is also covered: If ℓ_w is true under σ, then $v_{\ell_w} \in V'$; otherwise $v_C^{\ell_w} \in V'$. For every $x_i \in X_1$, the edge $\{v_i^1, v_i^2\}$ is covered by v_i^1, the edges $\{v_i^2, v_i^3\}$ and $\{v_i^3, v_{i+1}^2\}$ by v_i^3, and the edge $\{v_i^2, v_{i+1}^3\}$ by v_{i+1}^3. The edge $\{v_d, v_1^3\}$ is covered by v_1^3. Thus V' is a vertex cover of G_1 of size $3n_1 + 2m_1$. (end of the proof of Claim 3.18) $\diamond$

Claim 3.19. *Let $S_X \subseteq \{a_{x_i} \mid x_i \in X_1\}$ (resp. $\{a_{x_i} \mid x_i \in X_2\}$), $S^* \subseteq \{a_i^* \mid x_i \in X_1\}$ (resp. $\{a_i^* \mid x_i \in X_2\}$) and let $S_{\mathcal{C}} \subseteq \{a_{C_j^z} \mid C_j^z \in \varphi_1\}$ (resp. $\{a_{C_j^z} \mid C_j^z \in \varphi_2\}$). If the subgraph of G_1 (resp. G_2) induced by the edges of $S_X \cup S^* \cup S_{\mathcal{C}} \cup \{a_d\}$ admits a vertex cover of size at most $2|S_X| + |S^*| + 2|S_{\mathcal{C}}|$, then φ_1 (resp. φ_2) admits a satisfying assignment.*

In particular, if G_1 (resp. G_2) admits a vertex cover V' of size $3n_1+2m_1$ (resp. $3n_2+2m_2$), then φ_1 (resp. φ_2) admits a satisfying assignment.

Proof. We show the statement for G_1. The reasoning for G_2 is analogous: The graph is generated using the same gadgets as G_1 but over φ_2. Observe that for every clause $a_C \in S_{\mathcal{C}}$, we need at least two vertices to cover the edges owned by a_C. These vertices may also cover edges between literal vertices and clause vertices. However, for every variable agent $a_{x_i} \in S_X$, we need at least two further vertices from $\{v_{x_i}, v_{\overline{x}_i}, v_i^1\}$ to cover her edges. The selected vertices may cover the edge $\{v_i^1, v_i^2\}$.

In total, we need at least $2|S_{\mathcal{C}}| + 2|S_X|$ many vertices to cover the aforementioned edges, leaving us with at most $|S^*|$ vertices to cover the remaining edges. These are at least the edges $E^* \coloneqq \{\{v_1^3, v_d\}\} \cup \bigcup_{a_i^* \in S^*} \{\{v_i^2, v_i^3\}, \{v_i^3, v_{i+1}^2\}, \{v_i^2, v_{i+1}^3\}\}$. Moreover, for every $a_i^* \in S^*$, at least one of the vertices $\{v_i^2, v_i^3\}$ must be in V' to cover the edge $\{v_i^2, v_i^3\}$, meaning we need at least $|S^*|$ vertices to cover them all. Thus we must use precisely $|S^*|$ vertices to cover them—more precisely, for every $a_i^* \in S^*$, *exactly* one of the vertices $\{v_i^2, v_i^3\}$ is in V', and no further vertices, including v_d, from G^C are in V'.

We will now use this fact to show that every chain agent in N_1 must be in S^*. Assume, towards a contradiction, that this is not the case. Consider the subgraph $G^C \coloneqq G_1[E^*]$. First assume that $S^* \neq \emptyset$. Since not every chain agent in N_1 is in $S_{\mathcal{C}}$, there must be an agent $a_i^* \in S^*$ such that $a_{i+1}^* \notin S^*$ (where $n_z + 1 = 1$). By earlier reasoning, we have that either v_i^2 or v_i^3 is in V', but not both. Moreover, as $a_{i+1}^* \notin S^*$, we have that $v_{i+1}^2, v_{i+1}^3 \notin V'$. However, this implies that either the edge $\{v_i^2, v_{i+1}^3\}$ or $\{v_i^3, v_{i+1}^2\}$ is uncovered, a contradiction. Now assume that $S^* = \emptyset$. The set V' must cover the edge $\{v_1^3, v_d\}$ that belongs to a_d, but we know that $v_d \notin V'$ and since $a_1^* \notin S^*$ since S^* is empty, we have that $v_1^3 \notin V'$, a contradiction.

Thus, we must have that every chain agent in N_1 is in S^*. Recall that as $v_d \notin V'$, we must have by earlier reasoning that $v_1^3 \in V'$ and $v_1^2 \notin V'$. To cover the edge $\{v_1^2, v_2^3\}$ we need that $v_2^3 \in V'$ and $v_1^2 \notin V'$. By repeating this argument we obtain that for every $x_i \in X_1$, it holds that $v_i^3 \in V'$ and $v_i^2 \notin V'$. However, for every $x_i \in X_1$, we must still cover the edge $\{v_i^1, v_i^2\}$. As $v_i^2 \notin V'$, we must have that $v_i^1 \in V'$. Thus, the size of V' is at least $2n_1$. Moreover, for every variable agent a_{x_i} in S_X we still need at least one vertex from $\{v_{x_i}, v_{\overline{x}_i}\}$ to cover this edge, and for every clause agent $a_C \in S_{\mathcal{C}}$, where $C = (\ell_1 \vee \ell_2 \vee \ell_3)$, we need at least two agents from $\{v_C^{\ell_1}, v_C^{\ell_2}, v_C^{\ell_3}\}$ to cover the edges between these vertices. Thus $|V'| \geq 2n_1 + |S_X| + 2|S_{\mathcal{C}}|$. Moreover, we assume that $|V'| \leq |S^*| + 2|S_X| + 2|S_{\mathcal{C}}|$. As $|S^*| = n_1$, we obtain that

$$2n_1 + |S_X| + 2|S_{\mathcal{C}}| \leq n_1 + 2|S_X| + 2|S_{\mathcal{C}}| \quad \Longrightarrow$$

$$n_1 \leq |S_X| \quad \Longrightarrow \quad |S_X| = n_1.$$

In other words, for every $x_i \in X_1$, we have that $a_{x_i} \in S_X$.

As we still require $2|S_{\mathcal{C}}|$ vertices to cover the edges belonging to the agents in $S_{\mathcal{C}}$ and $2n_1$ vertices to cover the edges belonging to the chain agents, we can have at most n_1-many vertices in V' whose all incident edges—restricted to the subgraph induced by the edges owned by the agents in $S_{\mathcal{C}} \cup S_X \cup S^*$—belong to variable agents. This means that for every $x_i \in X_1$, we must have *exactly* one vertex from $\{v_{x_i}, v_{\overline{x}_i}\}$ in V'. Moreover, for every $C = (\ell_1 \vee \ell_2 \vee \ell_3) \in \varphi_1$ such that $a_C \notin S_{\mathcal{C}}$, we have that $v_C^{\ell_w} \notin V'$ for any $w \in [3]$.

Consider assignment $\sigma \colon X_1 \to \{\mathsf{T}, \mathsf{F}\}$ that assigns a variable $x \in X_1$ to true if $v_x \in V'$ and false otherwise. Assignment σ must be a satisfying assignment: Assume towards a contradiction that there is a clause $C = (\ell_1 \vee \ell_2 \vee \ell_3) \in \varphi_1$ that is not satisfied under σ. This means that $v_{\ell_w} \notin V'$ for any $w \in [3]$. If $a_C \notin S_{\mathcal{C}}$, then the edge $\{v_{\ell_w}, v_C^{\ell_w}\}$ is not covered for any $w \in [3]$, a contradiction. Thus assume $a_C \in S_{\mathcal{C}}$. As we can have at most two vertices from $\{v_C^{\ell_w} \mid w \in [3]\}$ in V', there must be $w \in [3]$ such that $v_C^{\ell_w} \notin V'$. But then the edge $\{v_{\ell_w}, v_C^{\ell_w}\}$ is not covered, a contradiction. (end of the proof of Claim 3.19) $\diamond$

We now show that I is a yes-instance of SAT-UNSAT if and only if $\boldsymbol{\alpha}$ is core-stable.

For the "only if" direction, assume I is a yes-instance of SAT-UNSAT.

Let us start by showing the following claim that restricts the blocking coalitions we must consider:

Claim 3.20. *No set of agents $S \subseteq \mathsf{N}_1 \setminus \{a_d\}$ (resp. $\mathsf{N}_2 \setminus \{a_d\}$) may block.*

Proof. We show the statement for N_1; the reasoning for N_2 is analogous: They use the same gadgets in the construction and the used counts remain unchanged. Let S_X be the set of variable agents in S, $S_{\mathcal{C}}$ be the set of clause agents in S, and S^* be the set of chain agents in S. We observe that by construction $\sum_{i \in S} \boldsymbol{\alpha}_i = 2|S_X| + 2|S_{\mathcal{C}}| + |S^*|$.

Moreover, we claim that $c(S) \geq 2|S_X| + 2|S_{\mathcal{C}}| + |S^*|$, i.e., the minimum vertex cover of $G_1[\bigcup_{i \in S} E_i]$ is at least size $2|S_X| + 2|S_{\mathcal{C}}| + |S^*|$. Let V' be a minimum vertex cover of $G_1[\bigcup_{i \in S} E_i]$. For every clause agent $a_C \in S_{\mathcal{C}}$, where $C = (\ell_1 \vee \ell_2 \vee \ell_3)$, there must be at least two vertices in V' from $\{v_C^{\ell_w} \mid w \in [3]\}$ to cover the clique formed by these vertices. Similarly, for every $a_{x_i} \in S_X$, there must be at least two vertices in V' to cover the clique $\{v_{x_i}, v_{\neg x_i}, v_i^1\}$. For every chain agent $a_i^* \in S^*$, we need at least one vertex to cover the edge $\{v_i^2, v_i^3\}$. As none of the above enumerated sets overlap, we must have that $|V'| \geq 2|S_X| + 2|S_{\mathcal{C}}| + |S^*|$. Thus $\sum_{i \in S} \boldsymbol{\alpha}_i = 2|S_X| + 2|S_{\mathcal{C}}| + |S^*| \leq c(S)$, as required. (end of the proof of Claim 3.20) $\diamond$

Since the graphs G_1 and G_2 are not connected to each other and no agent owns edges on both of them, by Observation 3.13, it is sufficient for us to show that $\boldsymbol{\alpha}$ restricted to each of these graphs is core-stable. We have four possible cases:

Case 1: A subset $S \subseteq \mathsf{N}_1$ blocks $\boldsymbol{\alpha}$. By Claim 3.20, we have that $a_d \in S$ and $\sum_{i \in S \setminus \{a_d\}} \boldsymbol{\alpha}_i \leq c(S \setminus \{a_d\})$. Since a_d is allocated cost 0, it holds that $\sum_{i \in S \setminus \{a_d\}} \boldsymbol{\alpha}_i = \sum_{i \in S} \boldsymbol{\alpha}_i$. Additionally, since the size of a minimum vertex cover cannot decrease by adding edges, we have that $c(S \setminus \{a_d\}) \leq c(S)$. Thus we obtain that $\sum_{i \in S} \boldsymbol{\alpha}_i = \sum_{i \in S \setminus \{a_d\}} \boldsymbol{\alpha}_i \leq c(S \setminus \{a_d\}) \leq c(S)$, a contradiction to S blocking.

Case 2: We have that $\boldsymbol{\alpha}$ restricted to G_1 is not a preimputation. Observe that $\sum_{i \in \mathsf{N}_1} \boldsymbol{\alpha}_i = 3n_1 + 2m_1$. As the previous case implies that $\sum_{i \in \mathsf{N}_1} \boldsymbol{\alpha}_i \leq c(\mathsf{N}_1)$, we must have that $3n_1 + 2m_1 < c(\mathsf{N}_1)$, i.e., a minimum vertex cover of G_1 is strictly larger than $3n_1 + 2m_1$. However, as φ_1 admits a satisfying assignment, the graph G_1 admits a vertex cover of size $3n_1 + 2m_1$ by Claim 3.18, a contradiction.

Case 3: A subset $S \subseteq \mathsf{N}_2$ blocks $\boldsymbol{\alpha}$. By Claim 3.20, we have that $a_d \in S$. Let S_X be the set of variable agents in S, $S_{\mathcal{C}}$ be the set of clause agents in S, and S^* be the set of chain agents in S. We have that $\sum_{i \in S} \boldsymbol{\alpha}_i = 2|S_X| + |S^*| + 2|S_{\mathcal{C}}| + 1$. Thus we must have that $G_2[\bigcup_{i \in S} E_i]$ admits a vertex cover V' of size at most $2|S_X| + |S^*| + 2|S_{\mathcal{C}}|$. By Claim 3.19, formula φ_2 is then satisfiable. This, however, contradicts the fact that I is a yes-instance of SAT-UNSAT.

Case 4: We have that $\boldsymbol{\alpha}$ restricted to G_2 is not a preimputation. The previous case establishes that $\sum_{i \in \mathsf{N}_2} \boldsymbol{\alpha}_i \leq c(\mathsf{N}_2)$, as otherwise N_2 would block. Thus we must have that $c(\mathsf{N}_2) > \sum_{i \in \mathsf{N}_2} \boldsymbol{\alpha}_i = 3n_2 + 2m_2 + 1$, i.e., a minimum vertex cover of G_2 is strictly larger than $3n_2 + 2m_2 + 1$. Towards a contradiction, we construct a vertex cover V' of G_2 of size $3n_2 + 2m_2 + 1$. Let $V' \coloneqq \{v_C^{\ell_1}, v_C^{\ell_2} \mid C = (\ell_1 \vee \ell_2 \vee \ell_3) \in \varphi_2\} \cup \{v_{x_i}, v_{\neg x_i} \mid x_i \in X_2\} \cup \{v_i^2 \mid x_i \in X_2\} \cup \{v_d\}$. Clearly $|V'| = 3n_2 + 2m_2 + 1$. It remains to show this is indeed a vertex cover. For every $C = (\ell_1 \vee \ell_2 \vee \ell_3) \in \varphi_2$, clearly $v_C^{\ell_1}$ and $v_C^{\ell_2}$ cover the edges belonging to a_C. For every $x_i \in X_2$, the edges connecting v_{x_i} and $v_{\neg x_i}$ to the clause vertices are clearly covered by them. The vertices v_{x_i} and $v_{\neg x_i}$ also cover the edges that connect them to v_i^1. All the edges belonging to a_{x_i} are covered. The edges $\{v_i^1, v_i^2\}$, $\{v_i^2, v_i^3\}$, and $\{v_i^2, v_{i+1}^3\}$ are covered by v_i^2, whereas the edge $\{v_i^3, v_{i+1}^2\}$ is covered by v_{i+1}^2. Thus, all the edges belonging to a_i^* are covered. Finally, the edge $\{v_d, v_i^3\}$ that belongs to a_d is covered by v_d.

This concludes the proof of the "only if" direction.

For the "if" direction, assume $\boldsymbol{\alpha}$ is core-stable. Because G_1 and G_2 are completely disjoint and no agent has edges on both graphs, we must have that $\boldsymbol{\alpha}$ is core-stable when restricted to either of these graphs by Observation 3.13.

We first show that φ_1 admits a satisfying assignment. Since $\sum_{i \in \mathsf{N}_1} \boldsymbol{\alpha}_i = 3n_1 + 2m_1$, the graph G_1 must admit a vertex cover V' of size $3n_1 + 2m_1$. By Claim 3.19, this implies that φ_1 admits a satisfying assignment.

Next, we show that φ_2 cannot admit a satisfying assignment. Since $\sum_{i \in \mathsf{N}_2} \boldsymbol{\alpha}_i = 3n_2 + 2m_2 + 1$, the graph G_2 cannot admit a vertex cover V' of size $3n_2 + 2m_2$. By the

contrapositive of Claim 3.18, the formula φ_2 cannot admit a satisfying assignment. Thus I is a yes-instance of SAT-UNSAT. This concludes the proof. □

3.3.2 Core-Stability Existence

We now move to consider PCOG-MinVC-CE. We first use the ellipsoid method, inspired by Biró et al. [BKPP19], to show Θ_2^{P}-containment.

Theorem 3.21. *PCOG-MinVC-CE is contained in* Θ_2^P.

Proof. Our proof will utilize the fact that the following hinted version of PCOG-MinVC-CE is in coNP, where the input additionally has the size of a minimum vertex cover:

Hinted-PCOG-MinVC-CE (Hinted-VC)
Input: An instance $G = (V, E = E_1 \cup \cdots \cup E_{\hat{n}})$ of PCOG-MinVC-CE and k^* which is the size of a minimum vertex cover of G.
Question: Is G a yes-instance of PCOG-MinVC-CE, i.e., does it admit a core-stable allocation?

We show that this problem is contained in coNP:

Claim 3.22. *Hinted-VC is contained in* coNP.

Proof. We show coNP-containment using the ellipsoid method; see Section 2.3.2 for more details. More specifically, we use the variant where the constraints are not given explicitly. Our proof proceeds in two parts: In part 1, we show that every no-instance of Hinted-VC admits a polynomial-size witness, and in part 2, we show how to verify the witness. The verification procedure utilizes the minimum vertex cover size k^*, which is given as part of the input. Note that without k^*, it is not possible to verify the witness and therefore our result does not imply that PCOG-MinVC-CE is in coNP.

Part 1. We start with an informal description: First, recall that there exists an LP formulation for every TU-game (cf. Section 2.4.2). Suppose we are given a separation oracle (cf. Definition 2.39) as a blackbox algorithm. Then, by Theorem 2.40, the ellipsoid method would solve the problem in polynomial time. In the case of a no-instance, the separation oracle calls always return violated constraints, which consist of a coalition of players S and the minimum vertex cover size for the graph induced by S. Informally, our witness will consist of a sequence of coalitions together with minimum vertex covers for the graph induced by these coalitions. These witnesses will replace our separation oracle calls, and thereby allow us to execute the ellipsoid method with all our separation oracle calls already being available in the witness.

Let (G, k^*) be a no-instance of Hinted-VC. Observe that an allocation $\boldsymbol{\alpha}$ is core-stable if and only if it satisfies the core stability constraints from Definition 2.51[6]. In other

[6]Section 2.4 also provides an explicit LP formulation.

words, $\boldsymbol{\alpha}$ is core-stable if and only if it belongs to the polytope

$$P \coloneqq \left\{ \boldsymbol{\alpha} \in \mathbb{R}^{\hat{n}}_{\geq 0} \;\middle|\; \sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = k^* \text{ and } \sum_{i \in S} \boldsymbol{\alpha}_i \leq c(S)\ \forall S \subseteq \mathsf{N} \right\}, \tag{3.1}$$

where $c(S)$ denotes the size of a minimum vertex cover of $G[\bigcup_{i \in S} E_i]$. Thus, (G, k^*) is a yes-instance of Hinted-VC if and only if $P \neq \emptyset$. The polytope P is given implicitly through the graph G, and therefore, we use the separation oracle version of the ellipsoid method.

Let Q be a separation oracle for P. Given an allocation $\boldsymbol{\alpha}$, oracle Q either certifies that $\boldsymbol{\alpha} \in P$, or returns a violated constraint of P. In other words,

- if $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i \neq k^*$, then Q returns the violated preimputation constraint $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = k^*$; and
- otherwise, if there is a coalition $S \subseteq \mathsf{N}$ with $c(S) < \sum_{i \in S} \boldsymbol{\alpha}_i$, then Q returns the violated coalition constraint $\sum_{i \in S} \boldsymbol{\alpha}_i \leq c(S)$, where $c(S)$ is the size of a minimum vertex cover of $G[\bigcup_{i \in S} E_i]$.

Since G does not admit a core-stable allocation, every query made by the ellipsoid method yields a violated constraint. Let

$$\mathcal{T} \coloneqq ((\boldsymbol{\alpha}^1, \omega^1), \ldots, (\boldsymbol{\alpha}^r, \omega^r))$$

denote the transcript of one execution of the ellipsoid method on (G, k^*), where $\boldsymbol{\alpha}^j$ is the j-th queried allocation and ω^j is the violated constraint returned by Q. By Theorem 2.40, the ellipsoid method queries the separation oracle a polynomial number of times, as both the dimension (the number of agents in our case) and the maximum encoding length of a constraint[7] are bounded by the input size. Hence, r is polynomially bounded in $|V|$, the number of vertices.

The transcript $\mathcal{T}$ is not directly verifiable in polynomial time: Checking that a coalition constraint ω^j, i.e., $\sum_{i \in S_j} \boldsymbol{\alpha}_i \leq c(S_j)$, is violated at $\boldsymbol{\alpha}^j$ requires the minimum vertex cover size $c(S_j)$, which is NP-hard to compute. In each step, we therefore attach a witness of the violation: a vertex cover V'_j of $G[\bigcup_{i \in S_j} E_i]$ with $|V'_j| < \sum_{i \in S_j} \boldsymbol{\alpha}^j_i$, which certifies $c(S_j) \leq |V'_j| < \sum_{i \in S_j} \boldsymbol{\alpha}^j_i$. As k^* is part of the input, a violated preimputation constraint is recognized directly by testing $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}^j_i = k^*$ and needs no witness; we henceforth assume that every ω^j is a coalition constraint. This yields the witness

$$\mathcal{W} \coloneqq ((S_1, V'_1), \ldots, (S_r, V'_r))$$

of size polynomial in $|V|$.

[7] Remember that the longest constraint is of the form $\sum_{i \in [\hat{n}]} \boldsymbol{\alpha}_i = k^*$.

Part 2. Assume we are given $\mathcal{W}$ as a witness that G does not admit a core-stable allocation. We verify $\mathcal{W}$ by simulating the deterministic ellipsoid method on (G, k^*), reconstructing the transcript $\mathcal{T}$ step by step. Whenever the simulation queries an allocation $\boldsymbol{\alpha}^j$, we first use the input value k^* to test whether $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}^j_i = k^*$; if not, the violated constraint ω^j is the preimputation constraint $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = k^*$. Otherwise, we take the entry (S_j, V'_j) of $\mathcal{W}$ and verify in polynomial time that V'_j is a vertex cover of $G[\bigcup_{i \in S_j} E_i]$ and that

$$|V'_j| < \sum_{i \in S_j} \boldsymbol{\alpha}^j_i.$$

This certifies $c(S_j) \leq |V'_j| < \sum_{i \in S_j} \boldsymbol{\alpha}^j_i$, so the coalition constraint $\omega^j = (\sum_{i \in S_j} \boldsymbol{\alpha}_i \leq c(S_j))$ is violated by $\boldsymbol{\alpha}^j$; that is, S_j blocks $\boldsymbol{\alpha}^j$. We supply ω^j to the simulated ellipsoid method. The verifier accepts if and only if the simulation concludes that P is empty. As each of the r steps takes polynomial time and r is polynomially bounded, the verifier runs in polynomial time. (end of the proof of Claim 3.22) $\diamond$

We are now ready to show Θ_2^{P}-containment for PCOG-MinVC-CE. Let $G = (V, E = E_1 \cup \cdots \cup E_{\hat{n}})$ be an instance of PCOG-MinVC-CE. We first use binary search on $[|V|]$ to find the size k^* of a minimum vertex cover of G. This requires $\log(|V|)$ many calls to an NP-oracle solving Vertex Cover. Now we make one further NP-oracle call to the *co-problem* of Hinted-VC with the input G, which by Claim 3.22 is contained in NP. If this call returns no, then G admits a core-stable allocation, and we return yes; otherwise, we return no.

Clearly, we return yes if and only if G admits a core-stable allocation. In total, we require $\log(|V|)$ NP-oracle calls, as required. $\square$

We complement the above result with Θ_2^{P}-hardness, thereby showing that PCOG-MinVC-CE is Θ_2^{P}-complete. Intuitively, we reduce from Vertex Cover Membership[8] and use four agents to encode this instance. Three agents form a triangle through their edges, and overlap with the fourth agent in a single vertex. Only if this vertex lies in a minimum vertex cover, the instance is core-stable.

Theorem 3.23. *PCOG-MinVC-CE is Θ_2^P-hard even when $\hat{n} = 4$.*

Proof. We reduce from the Θ_2^{P}-hard Vertex Cover Membership problem.

Let $I = (G = (V, E), v')$ be an instance of Vertex Cover Membership. We construct an instance I' of PCOG-MinVC-CE with four agents $\mathsf{N} = [4]$. The reduction is illustrated in Figure 3.2. Let the graph of the reduced instance be $\hat{G} := (V \cup \{u_1, u_2\}, E_1 \cup E_2 \cup E_3 \cup E_4)$, where $E_1 := \{\{v', u_1\}\}$, $E_2 := \{\{v', u_2\}\}$, $E_3 := \{\{u_1, u_2\}\}$, and $E_4 := E$.

We claim that I is a yes-instance of Vertex Cover Membership if and only if I' is a yes-instance of PCOG-MinVC-CE.

[8] See Section 2.7 for a definition.

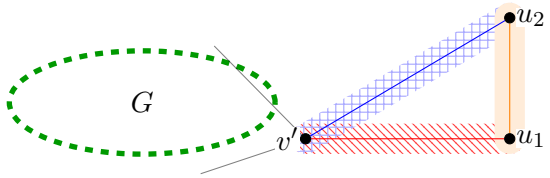


Figure 3.2: Illustration for the proof of Theorem 3.23. The connector vertex v' is shared by agents 1 and 2 (edges $\{v', u_1\}$ and $\{v', u_2\}$), the edge $\{u_1, u_2\}$ belongs to agent 3, and the dashed green ellipse marks agent 4's subgraph G. The lined red pattern indicates agent 1, the blue grid pattern agent 2, solid yellow agent 3, and the green dashed outline agent 4.

For the "only if" direction, assume I is a yes-instance of Vertex Cover Membership. Let k^* be the size of a minimum vertex cover of G and let V' be a vertex cover of G of size k^* that contains v'. We claim that the allocation $\boldsymbol{\alpha} = (0, 0, 1, k^*)$ is in the core.

First, observe that the minimum vertex cover of $\hat{G}$ is of size $k^* + 1$. For example, the set $V' \cup \{u_1\}$ is such a vertex cover: All the edges in E are clearly covered, as V' is a vertex cover of G. The edge $\{v', u_1\}$ is covered by both v' and u_1, the edge $\{v', u_2\}$ is covered by v', and the edge $\{u_1, u_2\}$ is covered by u_1. There cannot be a smaller vertex cover, as we need k^* vertices from V to cover the edges in E, leaving no vertices to cover the edge $\{u_1, u_2\}$. Thus $\boldsymbol{\alpha}$ is a preimputation.

To show $\boldsymbol{\alpha}$ is in the core, first observe that the minimum vertex cover of $\hat{G}[E_4]$ is k^*, and thus $\{4\}$ does not block $\boldsymbol{\alpha}$. Every subset $S \subseteq [3]$ satisfies $\sum_{i \in S} \boldsymbol{\alpha}_i \leq 1 \leq c(S)$, and thus cannot block $\boldsymbol{\alpha}$. For every $S \subseteq [2], S \neq \emptyset$ we have that $c(S \cup \{4\}) = k^* = \sum_{i \in S \cup \{4\}} \boldsymbol{\alpha}_i$ and $c(S \cup \{3, 4\}) = k^* + 1 = \sum_{i \in S \cup \{3,4\}} \boldsymbol{\alpha}_i$, as required. Therefore, $\boldsymbol{\alpha}$ is in the core, and I' is a yes-instance of PCOG-MinVC-CE.

For the "if" direction, assume I' is a yes-instance of PCOG-MinVC-CE. Let $\boldsymbol{\alpha} \in \mathbb{R}^4$ be in the core. Assume, towards a contradiction, that G does not admit a minimum vertex cover containing v'. Then the size of a minimum vertex cover of $\hat{G}$ is necessarily $k^* + 2$, where k^* is the size of a minimum vertex cover of G. This is because we need k^* vertices from $V \setminus \{v'\}$ to cover the edges in E, and two vertices from $\{v', u_1, u_2\}$ to cover the edges $\{v', u_1\}, \{v', u_2\}, \{u_1, u_2\}$. We have that $\boldsymbol{\alpha}_4 \leq k^*$, as otherwise $\{4\}$ would block. Thus $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 \geq k^* + 2 - k^* = 2$. Moreover, because a minimum vertex cover of $\hat{G}[E_1 \cup E_2]$ is of size 1, we have that $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 \leq 1 \implies \boldsymbol{\alpha}_3 \geq 1$. Because a minimum vertex cover of $\hat{G}[E_3]$ is of size 1, we also have that $\boldsymbol{\alpha}_3 \leq 1$, and thus $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 = \boldsymbol{\alpha}_3 = 1$. Therefore there must be $z \in [2]$ such that $\boldsymbol{\alpha}_z > 0 \implies \boldsymbol{\alpha}_z + \boldsymbol{\alpha}_3 > 1$. But we have that a minimum vertex cover of $\hat{G}[E_z \cup E_3]$ is of size 1, and thus $\boldsymbol{\alpha}_z + \boldsymbol{\alpha}_3 \leq 1$, a contradiction. This concludes the proof. □

The previous reduction has one large agent. Next, we show that PCOG-MinVC-CE remains Θ_2^{P}-hard even when the agent size is bounded by a constant. The central part of the reduction is inspired by the standard reduction from SAT to Vertex Cover.

However, we duplicate vertices and then add "prize" vertices to ensure that a minimum vertex cover of the graph corresponds to a satisfying assignment that minimizes the number of true variables. We also add some gadgets to ensure correctness.

Theorem 3.24. *PCOG-MinVC-CE is Θ_2^P-hard even when $c_{max} = 9$.*

Proof. **Informal description.** We first show that the problem MinAsg 2-SAT Member is Θ_2^{P}-hard. In this problem, we are given a 2-SAT formula, and the problem asks whether a given variable is set to true in a satisfying assignment that sets a minimum number of variables to true. We then show that PCOG-MinVC-CE is Θ_2^{P}-hard by reducing from MinAsg 2-SAT Member.

The idea of the proof is that we replace every vertex and edge of a reduction from SAT to Vertex Cover by *multi-vertices* (blocks of $6n$ vertices) and *multi-edges* (complete bipartite graphs between blocks of vertices). Due to multi-edges being complete bipartite graphs, a vertex cover must always contain all vertices in one of the multi-vertices that are endpoints of multi-edges. We then divide the multi-edges among multiple agents, such that no agent owns more than 9 edges.

We utilize four gadgets in the reduction:

- **Literal gadget.** To encode the fact that we need to assign a truth value for a satisfying assignment, we add a multi-edge between a positive and a negative literal multi-vertex for each variable x_i. In order to ensure that a satisfying assignment with a minimum number of variables set to true is chosen, we add a *prize vertex* attached to the negative side. This adds a cost of one unit for setting a variable to true, thereby ensuring the minimum vertex cover size reflects the number of true variables.
- **Clause gadget.** In order to ensure that a truth-assignment corresponding to the vertex cover satisfies every clause, we add a clause gadget. For each clause, the gadget consists of a multi-edge between two clause multi-vertices and their corresponding literal multi-vertices, enforcing that at least one literal per clause is set to true.
- **Chain gadget.** A problem in combinatorial optimization games is that one needs to consider every subset of agents as possible blocking coalitions. In order to circumvent this, we add a chain gadget. The chain gadget consists of two chains. A small chain connecting the agents of each variable, and a big chain connecting all variable and clause agents. The chain gadget ensures that every blocking coalition (for our utility vector) must include *almost* all agents for every variable and clause, as the chain gadgets add an additional penalty for missing agents.
- **Enforcer gadget.** In order to enforce that the PCOG-MinVC-CE instance is a no-instance if x_1 is not in a minimum satisfying assignment, we add an enforcer gadget. It consists of a triangle on three dedicated agents that can only be part of

a core allocation if x_1 is in a minimum satisfying assignment. Otherwise, the three agents directly lead to a blocking coalition.

Combining these gadgets leads to the following construction:

Formal proof. We start by showing that the following problem is Θ_2^{P}-hard:

MinAsg 2-SAT Member
Input: A 2-SAT formula φ over the set of variables X, a variable $x' \in X$.
Question: Does φ admit a satisfying assignment σ such that x' is true under σ and there is no satisfying assignment σ' such that σ' sets strictly fewer variables to true than σ?

Claim 3.25. *MinAsg 2-SAT Member is Θ_2^P-hard even if every variable appears at most twice positive and once negative.*

Proof. We reduce from Vertex Cover Membership. Let $(G = (V, E), v')$ be an instance of Vertex Cover Membership. Let us construct an assignment φ as follows: We create a variable x_i for every vertex $v_i \in V$. For every edge $\{v_i, v_j\} \in E$, we construct a clause $(x_i \vee x_j)$. For every subset $S \subseteq V$, let σ_S be an assignment that sets x_i to true for every $v_i \in S$, and x_j to false for every $v_j \in V \setminus S$. Clearly, we have that S is a vertex cover if and only if σ_S is a satisfying assignment. Moreover, the number of vertices in S is precisely the number of variables set true by σ_S. Finally, we have that $v' \in S$ if and only if σ_S sets x' to true. Thus I is a yes-instance of Vertex Cover Membership if and only if (φ, x') is a yes-instance of MinAsg 2-SAT Member.

Next, we show that we can bound the number of times each variable appears by strategies of Berman et al. [BKS03]. Let (φ, v') be an instance of MinAsg 2-SAT Member. Observe that by the previous construction, we can assume that every variable only appears as a positive literal. Let δ be the maximum number of times a literal appears in φ. Let us construct a modified instance (φ', v'') as follows: For every $x_i \in X, z \in [\delta]$, we construct a new variable x_i^z. Order the clauses where each variable appears arbitrarily. For each clause $C = (x_i \vee x_j)$, let z_i tell us which appearance of x_i and z_j which appearance of x_j this is. We add the clause $(x_i^{z_i} \vee x_j^{z_j})$ to φ'. Moreover, for every $x_i \in X, z \in [\delta]$, add the clause $(\overline{x}_i^z \vee x_i^{z+1})$, where $\delta + 1 = 1$. Observe that together these clauses are equivalent to $(x_i^1 \implies x_i^2 \implies \cdots \implies x_i^\delta \implies x_i^1)$ which means that $x_i^1 \iff x_i^2 \iff \cdots \iff x_i^\delta$, i.e., in any satisfying assignment, all the copies of x_i are assigned the same way. For the special vertex x', let the special vertex of the reduced instance be x'^1

It is easy to observe that every variable appears at most twice positive and once negative. Moreover, clearly every satisfying assignment of φ has a corresponding satisfying assignment of φ' and vice versa. Moreover, if a satisfying assignment σ of φ sets ℓ variables to true, clearly the corresponding satisfying assignment of σ' sets $\delta\ell$ variables to true. Finally, if a satisfying assignment sets x' to true, then the special variable of the reduced instance x'^1 must also be true. (end of the proof of Claim 3.25) $\diamond$

In what follows, we may call a set of $6n$-many vertices a *multi-vertex*. A *multi-edge* between two multi-vertices $U_1 = \{u_1^z \mid z \in [6n]\}$, $U_2 = \{u_2^z \mid z \in [6n]\}$ is the set of all possible edges between the sets U_1 and U_2, i.e., the set $\{\{u_1^z, u_2^w\} \mid (z, w) \in [6n] \times [6n]\}$.

The following observation shows that multi-vertices and multi-edges behave similarly to normal vertices and edges:

Observation 3.26. *Let G be a graph that contains multi-vertices U_1 and U_2 and a multi-edge between them. Then, for every vertex cover V' of G, it holds that $U_1 \subseteq V'$ or $U_2 \subseteq V'$.*

Proof. Assume, towards a contradiction that there is a vertex cover V' of G such that there are $u_1^z \in U_1$ and $u_2^w \in U_2$ such that $u_1^z, u_2^w \notin V'$. Then V' does not cover the edge $\{u_1^z, u_2^w\}$, a contradiction. (end of the proof of Observation 3.26) $\diamond$

Let $I = (\varphi, x')$ be an instance of MinAsg 2-SAT Member where every variable appears at most twice positive and once negative. As shown in Claim 3.25, this is Θ_2^{P}-hard. Let φ be over the set of variables $X = \{x_i \mid i \in [n]\}$, and let $m \coloneqq |\varphi|$. We may assume, without loss of generality, that $x' = x_1$. For notational convenience, let $Y = \varphi \cup X$. We let $Y = \{y_1, \ldots, y_{n+m}\}$, where for every $x_i \in [n]$, we name $y_i = x_i$, and for every $C_j \in \varphi$, we name $y_{n+j} = C_j$.

We construct an instance I' of PCOG-MinVC-CE with a graph G as follows: Let us first construct the vertices:

- For every variable $x_i \in X$, we construct:
  - a prize vertex v_i^*,
  - a positive literal multi-vertex $V_i = \{v_i^z \mid z \in [6n]\}$ and a negative literal multi-vertex $\bar{V}_i = \{\bar{v}_i^z \mid z \in [6n]\}$, and
  - for every $(z, w) \in [6n] \times [6n], t \in [3]$ a small chain vertex $b_{i,t}^{z,w}$.
- For every clause $C_j = (\ell_i \vee \ell_{i'}) \in \varphi$, we construct clause multi-vertices $U_{j,\ell_i} = \{u_{j,\ell_i}^z \mid z \in [6n]\}$ and $U_{j,\ell_{i'}} = \{u_{j,\ell_{i'}}^z \mid z \in [6n]\}$.
- Moreover, for every $y_g \in Y$, $t \in [3]$, we construct a big chain multi-vertex $D_{g,t} = \{d_{g,t}^z \mid z \in [6n]\}$.
- Finally we construct two enforcer vertices s_1 and s_2.

Let us now construct the edges:

- For every variable $x_i \in X$, we construct:
  - a multi-edge between the multi-vertices V_i and $\bar{V}_i$,

  - the edge $\{\bar{v}_i^1, v_i^*\}$,
  - for every $(z, w) \in [6n] \times [6n]$, we add the edges $\{b_{i,1}^{z,w}, b_{i,2}^{z,w}\}$ and $\{b_{i,3}^{z,w}, b_{i,1}^{\hat{z},\hat{w}}\}$, where $(\hat{z}, \hat{w})$ is $(z, w+1)$ if $(w < 6n)$, otherwise $(z+1, 1)$, where $6n + 1 = 1$.

- For every clause $C_j = (\ell_i \vee \ell_{i'}) \in \varphi$, for every $i'' \in \{i, i'\}$, we construct a multi-edge between $U_{j,\ell_{i''}}$ and $V_{i''}$ if $\ell_{i''}$ is a positive literal, otherwise between $U_{j,\ell_{i''}}$ and $\bar{V}_{i''}$.
- Moreover, for every $y_g \in Y$, we construct a multi-edge between the multi-vertices $D_{g,1}$ and $D_{g,2}$ and between the multi-vertices $D_{g,3}$ and $D_{g+1,1}$, where $n + m + 1 = 1$.
- Finally, we add the edges $\{s_1, s_2\}, \{s_1, v_1^1\}$, and $\{s_2, v_1^1\}$.

Let us now describe the set of agents N:

- For every variable $x_i \in X$, let C_{i_1}, C_{i_2} be the two clauses where x_i appears (if they exist), and let $C_{\bar{i}}$ be the clause where $\overline{x}_i$ appears. For every tuple $(z, w) \in [6n] \times [6n]$, we construct a variable agent $a_{x_i}^{z,w}$ who owns the following edges:
  - $\{u_{i_1,\ell_i}^z, v_i^w\}, \{u_{i_1,\ell_i}^z, v_i^w\}$, and $\{u_{\bar{i},\ell_i}^z, v_i^w\}$, if C_{i_1}, C_{i_2}, and $C_{\bar{i}}$ exist, respectively,
  - $\{v_i^w, \bar{v}_i^z\}$,
  - $\{d_{i,1}^z, d_{i,2}^w\}$, and $\{d_{i,3}^z, d_{i+1,1}^w\}$,
  - $\{b_{i,1}^{z,w}, b_{i,2}^{z,w}\}$ and $\{b_{i,3}^{z,w}, b_{i,1}^{\hat{z},\hat{w}}\}$, where $\hat{z}, \hat{w}$ is $z, w+1$ if $w < 6n$, otherwise $z+1, 1$, where $6n + 1 = 1$, and
  - if $z = w = 1$, then $\{\bar{v}_i^1, v_i^*\}$.
- For every clause $C_j = (\ell_i \vee \ell_{i'}) \in \varphi$, for every tuple $(z, w) \in [6n] \times [6n]$, we construct a clause agent $a_{C_i}^{z,w}$ who owns the following edges:
  - $\{u_{j,\ell_i}^z, u_{j,\ell_{i'}}^w\}$,
  - $\{d_{n+j,1}^z, d_{n+j,2}^w\}$ and $\{d_{n+j,3}^z, d_{n+j+1,1}^z\}$, where $n + m + 1 = 1$.
- We construct three enforcer agents $\hat{a}_1, \hat{a}_2$, and $\hat{a}_3$. Agent $\hat{a}_1$ owns the edge $\{s_1, v_1^1\}$, agent $\hat{a}_2$ owns the edge $\{s_2, v_1^1\}$, and agent $\hat{a}_3$ owns the edge $\{s_1, s_2\}$.

The variable agents own at most 9 edges, and the clause agents at most 3 edges. The enforcer agents each own an edge. Thus $c_{\max} = 9$.

Given an agent $a \in \mathsf{N}$, we use E_a to denote the set of edges a owns. For a variable or clause $y_g \in Y$, let $A_{y_g} \coloneqq \{a_{y_g}^{z,w} \mid (z, w) \in [6n] \times [6n]\}$. If $y_i = x_i$ is a variable, we define $A_{x_i} \coloneqq A_{y_i}$. Let $N^\varphi \coloneqq \bigcup_{y_g \in Y} A_{y_g}$.

We start by showing the connection between our construction and the original SAT-formula:

Claim 3.27. *1. If φ admits a satisfying assignment $\sigma\colon X \to \{\mathsf{T},\mathsf{F}\}$ that sets β variables to true, then $G[\bigcup_{a\in N^\varphi} E_a]$ admits a vertex cover V' of size $12n^2+36n^3+12mn+\beta$. Moreover, if $\sigma(x_1)=\mathsf{T}$, then V' contains v_1^1.*

2. If $G[\bigcup_{a\in N^\varphi} E_a]$ admits a minimal vertex cover V' of size $12n^2+36n^3+12mn+\beta$, then φ admits a satisfying assignment σ that sets at most β variables to true. Moreover, if V' contains v_1^1, then $\sigma(x_1)=\mathsf{T}$.

Proof. We start by showing Statement 1. We construct a vertex cover V' as follows: For every $x_i \in X$ such that $\sigma(x_i)=\mathsf{T}$, we add the vertices in $V_i \cup \{\bar{v}_i^1\}$ to V'. For every $x_i \in X$ such that $\sigma(x_i)=\mathsf{F}$, we add the vertices in $\bar{V}_i$ to V'. For every $x_i \in X$ we add the vertices in $\{b_{i,1}^{z,w} \mid (z,w)\in[6n]\times[6n]\}$ to V'. For every clause $C_j \in \varphi$, let $\ell_i \in C_j$ be a literal of C_j that is false under σ, or if no literal of C_j is false under σ, an arbitrary literal of C_j. Observe that since σ is a satisfying assignment, there is at most one literal in C_j that is false under σ. We add the vertices in U_{j,ℓ_i} to V'. For every $y_g \in Y$, we add $D_{g,1}$ to V'. Since σ sets β variables to true, we get that the size of the vertex cover V' is

$$\beta(6n+1)+(n-\beta)6n+n\cdot 36n^2+m\cdot 6n+(n+m)6n = 12n^2+36n^3+12mn+\beta,$$

as required.

It remains to show that V' is indeed a vertex cover. Clearly, for every $x_i \in X$, the edges that belong to the multi-edge between V_i and $\bar{V}_i$ are covered. The edge $\{\bar{v}_i^1, v_i^*\}$ is also covered, as $\bar{v}_i^1 \in V'$ regardless of whether σ assigns x_i to true or false. For every $C_j \in \varphi$ such that $x_i \in C_j$, if x_i is true under σ, then $V_i \subseteq V'$, otherwise $U_{j,x_i} \subseteq V'$. Thus the edges in the multi-edge between V_i and U_{j,x_i} are covered. Through similar argumentation all the edges in the multi-edge between $\bar{V}_i$ and $U_{j,\overline{x}_i}$ are covered for any $C_j \in \varphi$ that contains $\overline{x}_i$. For every $(z,w)\in[6n]\times[6n]$, the edge $\{b_{i,1}^{z,w}, b_{i,2}^{z,w}\}$ is covered by $b_{i,1}^{z,w}$, and the edge $\{b_{i,3}^{z,w}, b_{i,1}^{\hat{z},\hat{w}}\}$ by $b_{i,1}^{\hat{z},\hat{w}}$, where $(\hat{z},\hat{w})$ is $(z,w+1)$ if $w<6n$, otherwise $(z+1,1)$, where $6n+1=1$. For every $C_j=(\ell_i\vee\ell_{i'})\in\varphi$, the vertices in the multi-edge between U_{j,ℓ_i} and $U_{j,\ell_{i'}}$ are covered by either the vertices in U_{j,ℓ_i} or $U_{j,\ell_{i'}}$. For every $y_g\in Y$, the multi-edge between $D_{g,1}$ and $D_{g,2}$ is covered by the vertices in $D_{g,1}$, and the multi-edge between $D_{g,3}$ and $D_{g+1,1}$ is covered by $D_{g+1,1}$, where $n+m+1=1$. Thus V' is indeed a vertex cover. Moreover, if x_1 is true under σ, then clearly $v_1^1 \in V'$.

Next we show Statement 2. By Observation 3.26, for every $x_i \in X$ we need at least $6n$ vertices to cover the multi-edge between V_i and $\bar{V}_i$. Also observe that if $V_i \subseteq V'$, then we need an additional vertex to cover the edge $\{\bar{v}_i^1, v_i^*\}$. We also need at least $36n^2$ further vertices to cover the edges $\{b_{i,1}^{z,w}, b_{i,2}^{z,w}\}$ for every $(z,w)\in[6n]\times[6n]$. Similarly we need at least $6n$ vertices for every $C_j=(\ell_i\vee\ell_{i'})$ to cover the multi-edge between U_{j,ℓ_i} and $U_{j,\ell_{i'}}$. For every $y_g\in Y$, we need $6n$ vertices to cover the multi-edge between $D_{g,1}$ and $D_{g,2}$. This is in total $n\cdot 6n+36n^2+m\cdot 6n+(n+m)6n+\alpha = 12n^2+36n^3+12mn+\alpha$ vertices, where α is the number of variables $x_i \in X$ such that $V_i \subseteq V'$. Since $|V'| = 12n^2+36n^3+12mn+\beta$, we must have that $\alpha \le \beta$. Moreover, we can have at most $\beta-\alpha \le n$ vertices in V' that are not listed here.

Let us construct an assignment σ as follows: For every $x_i \in X$, if $V_i \subseteq V'$, then x_i is true under σ, otherwise it is false. Clearly, this sets $\alpha \leq \beta$ variables to true, as required. Assume, towards a contradiction, that σ is not a satisfying assignment. Then there is a clause $C_j = (\ell_i \vee \ell_{i'})$ such that neither of its literals is true under σ. Assume, without loss of generality, that $U_{j,\ell_i} \subseteq V'$. Also, assume that ℓ_i is the true literal x_i; if instead $\ell_i = \overline{x}_i$, swap V_i and $\bar{V}_i$: The multi-edge gadget (Observation 3.26) is identical, so the same $6n > n$ count gives the contradiction. Then we have that $\bar{V}_i \subseteq V'$. By earlier reasoning, we have at most n vertices left to cover the multi-edge between V_i and U_{j,ℓ_i}; however, this requires by Observation 3.26 at least $6n > n$ vertices, a contradiction. Thus σ must be a satisfying assignment.

Finally, we must show that if V' contains v_1^1, then σ sets x_1 to true. Assume, towards a contradiction, that this is not the case. Then $\{v_1^1\} \cup \bar{V}_1 \subseteq V'$. We show that $V' \setminus \{v_1^1\}$ is also a vertex cover, contradicting the minimality of V'. The edges in the multi-edge between V_1 and $\bar{V}_1$ are covered by $\bar{V}_1$, so they remain covered by $V' \setminus \{v_1^1\}$. Consider $C_j \in \varphi$ such that $x_1 \in C_j$. By Observation 3.26, it holds that $V_1 \subseteq V'$ or $U_{j,x_1} \subseteq V'$. Since $\bar{V}_i \subseteq V'$, there are at most n vertices in $V_i \cap V'$, meaning that $U_{j,x_i} \subseteq V'$. But then $V' \setminus \{v_1^1\}$ covers the multi-edge between U_{j,x_i} and V_1. Thus $V' \setminus \{v_1^1\}$ is also a vertex cover, as required. (end of the proof of Claim 3.27) $\diamond$

Now we can move on to prove that the two instances are equivalent, i.e., I is a yes-instance of MinAsg 2-SAT Member if and only if I' is a yes-instance of PCOG-MinVC-CE.

For the "only if" direction, assume I is a yes-instance of MinAsg 2-SAT Member. Let $\sigma \colon X \to \{\mathsf{T}, \mathsf{F}\}$ be a satisfying assignment such that $\sigma(x_1) = \mathsf{T}$ and there is no satisfying assignment that sets fewer variables to true. Let β be the number of variables that σ sets to true.

We construct an allocation $\boldsymbol{\alpha}$ as follows:

- For every $x_i \in X, (z, w) \in [6n] \times [6n]$, if $z = w = 1$, then we allocate $a_{x_i}^{z,w}$ the cost $\frac{1}{3n} + 1 + \frac{\beta}{n}$, otherwise we allocate $\frac{1}{3n} + 1$.
- For every $C_j \in \varphi, (z, w) \in [6n] \times [6n]$, we allocate agent $a_{C_i}^{z,w}$ the cost $\frac{1}{3n}$.
- The cost of $\hat{a}_1$ and $\hat{a}_2$ is 0, and the cost of $\hat{a}_3$ is 1.

Let us first show that $\boldsymbol{\alpha}$ is a preimputation, i.e., a minimum vertex cover of G has size $(n(\frac{1}{3n} + 1 + \frac{\beta}{n}) + (36n^2 - 1) \cdot n \cdot (\frac{1}{3n} + 1) + m \cdot 36n^2 \cdot \frac{1}{3n} + 1) = 12n^2 + 36n^3 + 12mn + \beta + 1$. By Claim 3.27.1, we have that $G[\bigcup_{a \in N^\varphi} E_a]$ admits a vertex cover V'' of size $12n^2 + 36n^3 + 12mn + \beta$, and $v_1^1 \in V''$. Thus the edges $\{v_1^1, s_1\}$ and $\{v_1^1, s_2\}$ are covered by V''. By adding one further vertex from $\{s_1, s_2\}$, we obtain a vertex cover of size $12n^2 + 36n^3 + 12mn + \beta + 1$.

Now assume towards a contradiction that G admits a vertex cover V' of size at most $12n^2 + 36n^3 + 12mn + \beta$. We need at least one vertex from $\{s_1, s_2\}$ in V'. Thus

$|V' \setminus \{s_1, s_2\}| \geq 12n^2 + 36n^3 + 12mn + \beta - 1$, and $V' \setminus \{s_1, s_2\}$ is a vertex cover of $G[\bigcup_{a \in N^\varphi} E_a]$ of size $12n^2 + 36n^3 + 12mn + \beta - 1$. By Claim 3.27.2, it holds that φ admits a satisfying assignment with $\beta - 1$ true variables, a contradiction.

Next, we show that $\boldsymbol{\alpha}$ is core-stable. Assume, towards a contradiction, that $\boldsymbol{\alpha}$ admits an inclusionwise minimal blocking coalition S. First we observe that $\hat{a}_1, \hat{a}_2, \hat{a}_3 \notin S$: In what follows, let γ be the size of a minimum vertex cover of $G[\bigcup_{a \in S} E_a]$ and γ' be the size of a minimum vertex cover of $G[\bigcup_{a \in S \setminus \{\hat{a}_1, \hat{a}_2, \hat{a}_3\}} E_a]$. If $\hat{a}_3 \in S$, then a minimum vertex cover of $G[\bigcup_{a \in S} E_a]$ must contain s_1 or s_2. Because s_1 and s_2 do not cover any edges belonging to the agents in $S \setminus \{\hat{a}_1, \hat{a}_2, \hat{a}_3\}$, we must have that $\gamma' \leq \gamma - 1$. Thus since S blocks $\boldsymbol{\alpha}$, we have that $c(S \setminus \{\hat{a}_1, \hat{a}_2, \hat{a}_3\}) = \gamma' \leq \gamma - 1 = c(S) - 1 < \sum_{a \in S} \boldsymbol{\alpha}_a - 1 = \sum_{a \in S \setminus \{\hat{a}_1, \hat{a}_2, \hat{a}_3\}} \boldsymbol{\alpha}_a + \sum_{a \in S \cap \{\hat{a}_1, \hat{a}_2, \hat{a}_3\}} \boldsymbol{\alpha}_a - 1 = \sum_{a \in S \setminus \{\hat{a}_1, \hat{a}_2, \hat{a}_3\}} \boldsymbol{\alpha}_a + 1 - 1 = \sum_{a \in S \setminus \{\hat{a}_1, \hat{a}_2, \hat{a}_3\}} \boldsymbol{\alpha}_a$. Thus $S \setminus \{\hat{a}_1, \hat{a}_2, \hat{a}_3\}$ is also a blocking coalition, and it is a subset of S, a contradiction.

Now assume that $\hat{a}_1 \in S$ or $\hat{a}_2 \in S$ and $\hat{a}_3 \notin S$. If $\hat{a}_1 \in S$, then the degree of s_1 in $G[\bigcup_{a \in S} E_a]$ is 1. Similarly, if $\hat{a}_2 \in S$, then the degree of s_2 is 1 in $G[\bigcup_{a \in S} E_a]$. Thus, there is a minimum vertex cover of $G[\bigcup_{a \in S} E_a]$ that does not contain them. Since the cost of both of these agents is 0, we can remove them from S and obtain a smaller blocking coalition.

We have now shown that S must consist of only variable and clause agents. For every $y_g \in Y$, let $\hat{k}(y_g) \coloneqq |A_{y_g} \cap S|$, i.e., the number of agents corresponding to this variable or clause that are in S. If y_g refers to a variable agent, we will use $\hat{k}(x_g) \coloneqq \hat{k}(y_g)$ to make it clear that we refer to an agent corresponding to a variable. Similarly, if y_g refers to a clause agent, we will use $\hat{k}(C_{g-n}) \coloneqq \hat{k}(y_g)$ to make it clear that we refer to an agent corresponding to a clause.

Now we can make the following observation about the cost allocated to the agents in S:

Observation 3.28. *It holds that* $\sum_{a \in S} \boldsymbol{\alpha}_a \leq \sum_{x_i \in X} \hat{k}(x_i)(\frac{1}{3n} + 1) + \gamma \frac{\beta}{n} + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n}$, *where γ is the number of variables that have at least one corresponding agent in S.*

Proof. Observe that for every $x_i \in X$ that has at least one corresponding agent in S, it holds that $\sum_{a \in A_{x_i}} \boldsymbol{\alpha}_a \leq \hat{k}(x_i)(\frac{1}{3n} + 1) + \frac{\beta}{n}$. For every $x_i \in X$ that does not have a corresponding agent in S, we trivially have that $\sum_{a \in A_{x_i}} \boldsymbol{\alpha}_a = 0 = 0(\frac{1}{3n} + 1) = \hat{k}(x_i)(\frac{1}{3n} + 1)$. For every $C_j \in \varphi$, we have that $\sum_{a \in A_{C_j}} \boldsymbol{\alpha}_a = \frac{\hat{k}(C_j)}{3n}$. The statement follows. (end of the proof of Observation 3.28) $\diamond$

Next, we observe some requirements of a minimum vertex cover of $G[\bigcup_{a \in S} E_a]$:

Claim 3.29. *Every minimum vertex cover V' of $G[\bigcup_{a \in S} E_a]$ must contain vertices from the following pairwise disjoint vertex sets:*

1. *for every* $x_i \in X$, *at least* $\lceil \frac{\hat{k}(x_i)}{6n} \rceil$ *vertices from* $V_i \cup \bar{V}_i$,

2. *for every* $C_j = (\ell_i \vee \ell_{i'}) \in \varphi$, *at least* $\lceil \frac{\hat{k}(C_j)}{6n} \rceil$ *vertices from* $U_{j,\ell_i} \cup U_{j,\ell_{i'}}$,

3. *for every* $x_i \in X$, *if* $A_{x_i} \subseteq S$, *then at least* $\hat{k}(x_i)$ *vertices from* $\{b_{i,t}^{z,w} \mid (z,w) \in [6n] \times [6n], t \in [3]\}$, *otherwise if* $A_{x_i} \cap S \neq \emptyset$, *then at least* $\hat{k}(x_i) + 1$ *vertices from* $\{b_{i,t}^{z,w} \mid (z,w) \in [6n] \times [6n], t \in [3]\}$, *and*

4. *for every* $y_g \in Y$, *at least* $\lceil \frac{\max(\hat{k}(y_g), \hat{k}(y_{g+1}))}{6n} \rceil$ *vertices from* $D_{g,3} \cup D_{g+1,1} \cup D_{g+1,2}$, *where* $n + m + 1 = 1$.

Proof. We first show Statement 1. The edges from multi-edge between V_i and $\bar{V}_i$ that are owned by agents in S must be covered. A vertex from V_i or $\bar{V}_i$ is incident to precisely $6n$ edges from this multi-edge, and can thus cover at most $6n$ edges, meaning that we need $\lceil \frac{\hat{k}(x_i)}{6n} \rceil$ vertices from $(V_i \cup \bar{V}_i)$. The proof of Statement 2 is analogous: The edges from multi-edge between U_{j,ℓ_i} and $U_{j,\ell_{i'}}$ that are in $G[\bigcup_{a \in S} E_a]$ must be covered, and minimum vertex cover of $G[\bigcup_{a \in S} E_a]$ must contain at least $\lceil \frac{\hat{k}(C_j)}{6n} \rceil$ vertices from $(U_{j,\ell_i} \cup U_{j,\ell_{i'}})$.

Now we show Statement 3. For every $(z,w) \in [6n] \times [6n]$ such that $a_{x_i}^{z,w} \in S$, we need at least one vertex from the edge $\{b_{i,1}^{z,w}, b_{i,2}^{z,w}\}$ in V'. In total, we need at least $\hat{k}(x_i)$ vertices. Moreover, if there is at least one $(z,w) \in [6n] \times [6n]$ such that $a_{x_i}^{z,w} \notin S$ and $A_{x_i} \cap S \neq \emptyset$, then there must be at least one $(\hat{z},\hat{w}) \in [6n] \times [6n]$ such that $a_{x_i}^{\hat{z},\hat{w}} \in S$ and the edge $\{b_{i,3}^{\hat{z},\hat{w}}, b_{i,1}^{z,w}\}$ exists in G. Then at least one of these vertices in $\{b_{i,3}^{\hat{z},\hat{w}}, b_{i,1}^{z,w}\}$ must be in a minimum vertex cover of $G[\bigcup_{a \in S} E_a]$, and this vertex is not counted in the earlier $\hat{k}(x_i)$-many vertices.

Finally, we show Statement 4. The edges from the multi-edge between $D_{g,3}$ and $D_{g+1,1}$ must be covered. These edges belong to the agents in A_{y_g}, and thus there are $\hat{k}(y_g)$ edges that need to be covered. A vertex from $D_{i,3} \cup D_{i+1,1}$ is incident to precisely $6n$ edges from this multi-edge, and thus we need at least $\lceil \frac{\hat{k}(y_g)}{6n} \rceil$ vertices from $D_{g,3} \cup D_{g+1,1}$ in V'. Similarly, the edges from the multi-edge between $D_{g+1,1}$ and $D_{g+1,2}$ must be covered. These edges belong to the agents in $A_{y_{g+1}}$, and thus we need at least $\lceil \frac{\hat{k}(y_{g+1})}{6n} \rceil$ vertices from $D_{g+1,1} \cup D_{g+1,2}$ in V'. Thus, we need at least $\lceil \frac{\max(\hat{k}(y_g), \hat{k}(y_{g+1}))}{6n} \rceil$ vertices from $D_{g,3} \cup D_{g+1,1} \cup D_{g+1,2}$ in V'. (end of the proof of Claim 3.29) $\diamond$

By adding together all the sets from the previous claim, we obtain the following two useful observations:

Observation 3.30. *It holds that*

$$c(S) \geq \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{6n} + 1) + \gamma' + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{6n} + \sum_{y_g \in Y} \frac{\max(\hat{k}(y_g), \hat{k}(y_{g+1}))}{6n}$$

$$\geq \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{3n} + 1) + \gamma' + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n},$$

where $y_{n+m+1} = y_1$ and γ' is the number of variables in X who have at least one and at most $36n^2 - 1$ corresponding agents in S.

Proof. By combining Claim 3.29(1–4), we obtain the first inequality. Observe that we can remove the ceiling functions, because they can only decrease the cost on the right-hand side of the bound. The second inequality is obtained by lower-bounding the cost of every max-function with its first argument. (end of the proof of Observation 3.30) ⋄

Claim 3.31. *For every minimum vertex cover V' of $G[\bigcup_{a \in S} E_a]$, the following holds:*

1. *For every $x_i \in X$, we have that $|(V_i \cup \bar{V}_i) \cap V'| \leq 6n + \beta$.*
2. *For every $C_j = (\ell_1 \vee \ell_2) \in \varphi$, we have that $|(U_{j,\ell_1} \cup U_{j,\ell_2}) \cap V'| \leq 6n + \beta$.*

Proof. We first show that for every $x_i \in X$, $|(V_i \cup \bar{V}_i) \cap V'| \leq \lceil \frac{\hat{k}(x_i)}{6n} \rceil + \beta$. As $\hat{k}(x_i) \leq 36n^2$, the statement then follows directly. Assume, towards a contradiction, that $|(V_i \cup \bar{V}_i) \cap V'| > \lceil \frac{\hat{k}(x_i)}{6n} \rceil + \beta$. By combining the statements of Claim 3.29 and utilizing that $\lceil x \rceil \geq x$ and $\max(x, y) \geq x$ for every $x, y > 0$, we obtain that $|V'| \geq \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{3n} + 1) + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n} + \beta$. From Observation 3.28 it follows that $\sum_{a \in S} \boldsymbol{\alpha}_a \leq \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{3n} + 1) + \beta + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n}$. However, this implies that $c(S) = |V'| \geq \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{3n} + 1) + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n} + \beta \geq \sum_{a \in S} \boldsymbol{\alpha}_a$, a contradiction to S blocking. Thus $|(V_i \cup \bar{V}_i) \cap V'| \leq \lceil \frac{\hat{k}(x_i)}{6n} \rceil + \beta$. Since $\hat{k}(x_i) \leq 36n^2$, the statement follows.

The proof for the second statement is analogous: Replace the variable multi-vertices $(V_i \cup \bar{V}_i)$ by the clause multi-vertices $(U_{j,\ell_1} \cup U_{j,\ell_2})$ and use part (2) of Claim 3.29 in place of part (1). (end of the proof of Claim 3.31) ⋄

We can now show that there must be at least one variable whose corresponding agents are all in S.

Claim 3.32. *There is $x_{\hat{i}} \in X$ such that $A_{x_{\hat{i}}} \subseteq S$.*

Proof. Assume, towards a contradiction, that no variable has all of its corresponding agents in S. Let γ' be the number of variables such that neither all nor none of the agents corresponding to this variable are in S, and γ is the number of variables that have

at least one agent in S. We must have that $\gamma' = \gamma$. By Observations 3.28 and 3.30, we obtain that

$$c(S) - \sum_{a \in S} \boldsymbol{\alpha}_a \geq \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{3n} + 1) + \gamma + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n} - \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{3n} + 1) - \gamma\frac{\beta}{n} - \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n} \geq \gamma - \gamma\frac{\beta}{n} \geq 0,$$

where the last inequality follows from the fact that $\beta \leq n$ by definition. Thus $c(S) \geq \sum_{a \in S} \boldsymbol{\alpha}_a$, which contradicts S blocking. (end of the proof of Claim 3.32) $\diamond$

Now we are ready to show the following:

Claim 3.33. *For every $y_g \in Y$, it holds that $\hat{k}(y_g) > 36n^2 - 6n\beta$.*

Proof. Let γ' be the number of variables such that neither all nor none of the agents corresponding to this variable are in S, and γ is the number of variables that have at least one agent in S. Since S is a blocking coalition, by Observations 3.28 and 3.30, and the fact that $0 \leq \gamma' \leq \gamma \leq n$, we obtain that

$$\begin{aligned}
0 >& c(S) - \sum_{a \in S} \boldsymbol{\alpha}_a \\
&\geq \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{6n} + 1) + \gamma' + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{6n} + \sum_{y_g \in Y} \frac{\max(\hat{k}(y_g), \hat{k}(y_{g+1}))}{6n} \\
&\quad - \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{3n} + 1) - \gamma\frac{\beta}{n} - \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n} \\
&\geq - \sum_{x_i \in X} \frac{\hat{k}(x_i)}{6n} - \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{6n} + \sum_{y_g \in Y} \frac{\max(\hat{k}(y_g), \hat{k}(y_{g+1}))}{6n} - \beta \\
&\geq \sum_{y_g \in Y} \frac{\max(\hat{k}(y_g), \hat{k}(y_{g+1})) - \hat{k}(y_g)}{6n} - \beta = \sum_{\substack{y_g \in Y \\ \hat{k}(y_{g+1}) > \hat{k}(y_g)}} \frac{\hat{k}(y_{g+1}) - \hat{k}(y_g)}{6n} - \beta,
\end{aligned}$$

where $y_{n+m+1} = y_1$.

Let $y_{\min} \coloneqq \arg\min_{y_g \in Y} \hat{k}(y_g)$ and let $y_{\max} \coloneqq \arg\max_{y_g \in Y} \hat{k}(y_g)$. Observe that we must have that $\sum_{\substack{y_g \in Y \\ \hat{k}(y_{g+1}) > \hat{k}(y_g)}} \frac{\hat{k}(y_{g+1}) - \hat{k}(y_g)}{6n} \geq \frac{\hat{k}(y_{\max}) - \hat{k}(y_{\min})}{6n}$. By Claim 3.32, there is $x_{\hat{i}} \in X$ such that $\hat{k}(x_{\hat{i}}) = 36n^2$, and thus $\hat{k}(y_{\max}) = 36n^2$. We obtain that

$$0 > \frac{36n^2 - \hat{k}(y_{\min})}{6n} - \beta \iff \hat{k}(y_{\min}) > 36n^2 - 6n\beta.$$

As $y_{\min} \coloneqq \arg\min_{y_g \in Y} \hat{k}(y_g)$, it holds that for every $y_g \in Y$ that $\hat{k}(y_g) > 36n^2 - 6n\beta$, as required. (end of the proof of Claim 3.33) $\diamond$

Let V'_S be a minimum vertex cover of S. We proceed to build an assignment $\sigma' \colon X \to \{\mathsf{T}, \mathsf{F}\}$ from S as follows: For every $x_i \in X$, we set $\sigma'(x_i) = \mathsf{T}$ if and only if $|V_i \cap V'_S| \geq |\bar{V}_i \cap V'_S|$.

We first show that σ' must be a satisfying assignment. Assume, towards a contradiction, that σ' is not a satisfying assignment. Then there must be a clause $C_j = (\ell_i \vee \ell_{i'}) \in \varphi$ such that neither of its literals is true under σ'. We must have that $|U_{j,\ell_i} \cap V'_S| \leq |U_{j,\ell_{i'}} \cap V'_S|$ or $|U_{j,\ell_i} \cap V'_S| \geq |U_{j,\ell_{i'}} \cap V'_S|$. Assume, without loss of generality, that $|U_{j,\ell_i} \cap V'_S| \leq |U_{j,\ell_{i'}} \cap V'_S|$. By Claim 3.31.2, we have that $|(U_{j,\ell_i} \cup U_{j,\ell_{i'}}) \cap V'_S| \leq 6n + \beta$ and thus $|U_{j,\ell_i} \cap V'_S| \leq \frac{6n+\beta}{2}$. Also, assume that ℓ_i is the true literal x_i; if instead $\ell_i = \overline{x}_i$, swap V_i and $\bar{V}_i$: The multi-edge gadget (Observation 3.26) is identical, so the same $6n > n$ count gives the contradiction. Since C_j is not satisfied under σ', we must have that $|V_i \cap V'_S| < |\bar{V}_i \cap V'_S|$. By Claim 3.31.1, we have that $|(V_i \cup \bar{V}_i) \cap V'_S| \leq 6n + \beta$ and thus $|V_i \cap V'_S| \leq \frac{6n+\beta}{2}$.

Recall that the multi-edge between V_i and U_{j,ℓ_i} consists of $36n^2$ edges, which are all owned by different agents in A_{x_i}. Since V'_S is a vertex cover of $G[\bigcup_{a \in S} E_a]$, each of these edges must be covered by V'_S or not be owned by an agent in S. The edges that are not covered by V'_S are those between the vertices in $V_i \setminus V'_S$ and $U_{j,\ell_i} \setminus V'_S$. Since there is an edge between every pair of vertices in the two aforementioned sets, we obtain that there are strictly more than

$$(6n - \frac{6n+\beta}{2})^2 \geq (\frac{2 \cdot 6n - 6n - n}{2})^2 = \frac{25n^2}{4}$$

edges that are not covered by V'_S. The agents who own these edges cannot be in S since V'_S is a vertex cover of $G[\bigcup_{a \in S} E_a]$. Since $\hat{k}(x_i) > 36n^2 - 6n\beta$ by Claim 3.33, there are strictly fewer than

$$6n\beta \leq 6n^2 < \frac{25n^2}{4}$$

edges in the multi-edge between V_i and U_{j,ℓ_i} that are not owned by agents in S. Thus, the edge set of $G[\bigcup_{a \in S} E_a]$ must contain edges that are not covered by V'_S, a contradiction to V'_S being a vertex cover.

We now have that σ' must be a satisfying assignment. Let us use this fact to compute a lower bound for the size of V'_S. Let β' be the number of variables σ' sets to true. Recall that no satisfying assignment can set fewer than β variables to true. Because σ' is a satisfying assignment, we have that $\beta' \geq \beta$.

- For every $x_i \in X$ such that $\sigma'(x_i) = \mathsf{T}$, we have two cases:

  Case 1: $A_{x_i} \subseteq S$. Every edge in the multi-edge between V_i and $\bar{V}_i$ and the edge $\{\bar{v}_i^1, v_i^*\}$ are owned by agents in S. By Observation 3.26, we must have that $V_i \subseteq V'_S$ or $\bar{V}_i \subseteq V'_S$. Since σ' sets x_i to true, we have that $|V_i \cap V'_S| \geq |\bar{V}_i \cap V'_S|$.

Therefore, if $\bar{V}_i \subseteq V'_S$, then $V_i \subseteq V'_S$. Thus $V_i \subseteq V'_S$ in either case. The vertices in V_i cannot cover the edge $\{\bar{v}_i^1, v_i^*\}$, so we need at least one vertex from this edge in V'_S. Thus we need at least $6n+1 = \lceil \frac{\hat{k}(x_i)}{6n} \rceil + 1$ vertices from $V_i \cup \bar{V}_i \cup \{v_i^*\}$ in V'_S. By Claim 3.29.3, we need further at least $\hat{k}(x_i)$ vertices from $\{b_{i,t}^{z,w} \mid (z,w) \in [6n] \times [6n], t \in [3]\}$.

Case 2: $A_{x_i} \nsubseteq S$. By Claim 3.33, there is at least one agent corresponding to x_i in S. Thus by Claim 3.29.3, the set V'_S must contain at least $\hat{k}(x_i)+1$ vertices from $\{b_{i,t}^{z,w} \mid (z,w) \in [6n] \times [6n], t \in [3]\}$. By Claim 3.29.1, we need further at least $\lceil \frac{\hat{k}(x_i)}{6n} \rceil$ vertices from $V_i \cup \bar{V}_i$.

In both cases we need $\lceil \frac{\hat{k}(x_i)}{6n} \rceil + \hat{k}(x_i) + 1$ vertices from $V_i \cup \bar{V}_i \cup \{v_i^*\} \cup \{b_{i,t}^{z,w} \mid (z,w) \in [6n] \times [6n], t \in [3]\}$ in V'_S.

- For every variable $x_i \in X$ such that $\sigma'(x_i) = \mathsf{F}$, we need $\lceil \frac{\hat{k}(x_i)}{6n} \rceil$ vertices from $V_i \cup \bar{V}_i$ by Claim 3.29.1 and at least $\hat{k}(x_i)$ vertices from $\{b_{i,t}^{z,w} \mid (z,w) \in [6n] \times [6n], t \in [3]\}$ by Claim 3.29.3.

- For every clause $C_j = (\ell_i \vee \ell_{i'}) \in \varphi$, we obtain from Claim 3.29.2 that we need at least $\lceil \frac{\hat{k}(C_j)}{6n} \rceil$ vertices from $U_{j,\ell_i} \cup U_{j,\ell_{i'}}$.

- For every $y_g \in Y$, by Claim 3.29.4, we need at least $\lceil \frac{\hat{k}(y_g)}{6n} \rceil$ from $D_{g,3} \cup D_{g+1,1} \cup D_{g+1,2}$, where $n+m+1 = 1$.

By combining the above, we have that

$$\begin{aligned} c(S) &= |V'_S| \\ &\geq \sum_{\substack{x_i \in X \\ \sigma'[x_i]=\mathsf{T}}} (\lceil \frac{\hat{k}(x_i)}{6n} \rceil + \hat{k}(x_i) + 1) + \sum_{\substack{x_i \in X \\ \sigma'[x_i]=\mathsf{F}}} (\lceil \frac{\hat{k}(x_i)}{6n} \rceil + \hat{k}(x_i)) + \\ &\quad \sum_{C_j \in \varphi} \lceil \frac{\hat{k}(C_j)}{6n} \rceil + \sum_{y_g \in Y} \lceil \frac{\hat{k}(y_g)}{6n} \rceil \\ &\geq \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{3n} + 1) + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n} + \beta' \end{aligned}$$

By Observation 3.28 and the fact that $\beta \leq \beta'$, we obtain $\sum_{a \in S} \boldsymbol{\alpha}_a \leq \sum_{x_i \in X} \hat{k}(x_i) \cdot (\frac{1}{3n} + 1) + \beta + \sum_{C_j \in \varphi} \frac{\hat{k}(C_j)}{3n} \leq c(S)$, a contradiction to S blocking. This shows that $\boldsymbol{\alpha}$ cannot admit a blocking coalition and must therefore be core-stable.

For the "if" direction, assume that the constructed instance I' admits a core-stable allocation $\boldsymbol{\alpha}$. Let k^* be the size of a minimum vertex cover of $G[\bigcup_{a\in N^\varphi} E_a]$. Assume first that $G[\bigcup_{a\in N^\varphi} E_a]$ does not admit a minimum vertex cover containing v_1^1. Then the size of a minimum vertex cover of G is necessarily k^*+2. This is because we need k^* vertices from $N^\varphi \setminus \{v_1^1\}$ to cover the edges belonging to the agents in N^φ, and two vertices from $\{v_1^1, s_1, s_2\}$ to cover the triangle between them. We have that $\sum_{a\in \mathsf{N}\setminus\{\hat{a}_1,\hat{a}_2,\hat{a}_3\}} \boldsymbol{\alpha}_a \leq k^*$, as otherwise $\mathsf{N}\setminus\{\hat{a}_1,\hat{a}_2,\hat{a}_3\}$ would block. Thus $\boldsymbol{\alpha}_{\hat{a}_1}+\boldsymbol{\alpha}_{\hat{a}_2}+\boldsymbol{\alpha}_{\hat{a}_3}\geq 2$. Since a minimum vertex cover of the edges of $\hat{a}_1$ and $\hat{a}_2$ is of size 1, we have that $\boldsymbol{\alpha}_{\hat{a}_1}+\boldsymbol{\alpha}_{\hat{a}_2}\leq 1$, and thus $\boldsymbol{\alpha}_{\hat{a}_3}\geq 1$. There must be $z\in[2]$ such that $\boldsymbol{\alpha}_{\hat{a}_z}>0$, which leads to $\boldsymbol{\alpha}_{\hat{a}_z}+\boldsymbol{\alpha}_{\hat{a}_3}>1$. But the minimum vertex cover of the edges of $\hat{a}_z$ and $\hat{a}_3$ is of size 1, a contradiction. Thus $G[\bigcup_{a\in N^\varphi} E_a]$ must admit a minimum vertex cover containing v_1^1.

By Claim 3.27.2, since $G[\bigcup_{a\in N^\varphi} E_a]$ admits a minimum vertex cover of size k^* that contains v_1^1, we have that φ admits a satisfying assignment that sets at most $k^*-12n^2+36n^3+12mn$ variables to true, including x_1. If φ admits a satisfying assignment which sets strictly fewer than $k^*-12n^2+36n^3+12mn$ variables to true, then by Claim 3.27.1, the graph $G[\bigcup_{a\in N^\varphi} E_a]$ admits a minimum vertex cover that is strictly smaller than k^*, a contradiction to k^* being the size of a minimum vertex cover of $G[\bigcup_{a\in N^\varphi} E_a]$. Thus φ must admit a satisfying assignment σ such that x_1 is true under σ and there is no satisfying assignment that sets fewer variables to true. This concludes the proof. □

3.4 Partitioned Minimum Dominating Set Game

We now move on to study partitioned minimum dominating set games. Their computational complexities behave similarly to those for partitioned minimum vertex cover games.

3.4.1 Core-Stability Verification

As already discussed for the case of minimum vertex cover, verifying whether a given allocation is core-stable reduces to addressing an NP- and a coNP-problem. Indeed, this problem is DP-complete. This proof is similar to the proof for vertex cover; see Theorem 3.16.

Theorem 3.34. *PCOG-MinDS-CV is* DP*-complete. The hardness remains even if* $\hat{n}=1$ *or* $c_{max}=4$.

Proof. The proof proceeds in three steps. First, we show DP-membership, and afterward, we present a DP-hardness reduction with only one agent. We afterward show the case where $c_{\max}=4$.

DP**-membership.** By Papadimitriou [Pap94], we need to provide two NP-problems P and Q, and show that PCOG-MinDS-CV $=P\cap\overline{Q}$, in other words, every instance

of PCOG-MinDS-CV is a yes-instance if and only if it is a *yes*-instance of P but a *no*-instance of Q. To this end, let P and Q be the following two NP-problems.

Problem P**:** Given an instance $I = (G = (V_1 \cup \cdots \cup V_{\hat{n}}, E), \boldsymbol{\alpha})$, does G admit a dominating set of size at most $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i$?

Problem Q**:** Given an instance $I = (G = (V_1 \cup \cdots \cup V_{\hat{n}}, E), \boldsymbol{\alpha})$, is there a subset $S \subseteq \mathsf{N}$ of agents such that the induced graph $G[\cup_{i \in S} V_i]$ has a dominating set with size strictly smaller than $\sum_{i \in S} \boldsymbol{\alpha}_i$?

It is straightforward to check that both problems are contained in NP since the witness of either problem has polynomial size and can be verified in polynomial time as well.

It remains to show that for each instance $I = (G = (V_1 \cup \cdots \cup V_{\hat{n}}, E), \boldsymbol{\alpha})$ of PCOG-MinDS-CV $\boldsymbol{\alpha}$ is core-stable if and only if I is a yes-instance of P and a no-instance of Q. By definition, allocation $\boldsymbol{\alpha}$ is core-stable if and only if $\boldsymbol{\alpha}$ is a preimputation and no subset of agents is blocking $\boldsymbol{\alpha}$. The latter is equivalent to I being a yes-instance of P and no subset of agents having a dominating set with size that is strictly smaller than their cost under the allocation $\boldsymbol{\alpha}$ (i.e., instance I being a no-instance of Q), as desired.

DP**-hardness when** $\hat{n} = 1$**.** To show DP-hardness for the restricted case where there is only one agent, we reduce from the DP-complete SAT-UNSAT problem; see Section 2.7. Intuitively, we encode satisfiability in an instance of Dominating Set in a way that an unsatisfiable formula requires one more vertex in a dominating set than a satisfiable one. Then we set the allocation in such a way that it is a preimputation if and only if the instance of SAT-UNSAT is a yes-instance.

Let (φ_1, φ_2) be an instance of SAT-UNSAT, where φ_1 and φ_2 are over the variable sets X_1 and X_2, respectively. We create an instance $I' = (G, \boldsymbol{\alpha})$ of PCOG-MinDS-CV with only one agent. The graph G consists of three disjoint subgraphs: G_1, $G_2^{(1)}$, and $G_2^{(2)}$, where G_1 corresponds to φ_1 and the remaining two $G_2^{(1)}$ and $G_2^{(2)}$ to φ_2. We note that each subgraph is a slight modification of the standard reduction from 3SAT to Dominating Set.

Graph G_1*.* We create the following vertices for G_1:

- For every variable $x \in X_1$, create a positive literal vertex v_x, a negative literal vertex $v_{\overline{x}}$, and a dummy vertex d_x.
- For every clause $C \in \varphi_1$, create a clause vertex v_C.
- Create a special vertex v^*.

We create the following edges for G_1:

- The three vertices corresponding to the same variable form a triangle: For every $x \in X_1$, add the edges $\{v_x, v_{\overline{x}}\}$, $\{v_x, d_x\}$, and $\{v_{\overline{x}}, d_x\}$.
- Each clause vertex is adjacent to the vertices that correspond to the literals in the clause: For every clause $C \in \varphi_1$ and every literal $\ell \in C$, add edge $\{v_C, v_\ell\}$.
- The special vertex is adjacent to all clause and literal vertices, except the dummies: For every $v_y \in \{v_x, v_{\overline{x}} \mid x \in X_1\} \cup \{v_C \mid C \in \varphi_1\}$, we add edge $\{v^*, v_y\}$.

Graphs $G_2^{(1)}$ and $G_2^{(2)}$. These two graphs are constructed in the same way as G_1, except that φ_2 and X_2 are the underlying CNF formula and variable set.

For the sake of brevity, define $n_1 \coloneqq |X_1|$ and $n_2 \coloneqq |X_2|$. Finally, we construct the value vector $\boldsymbol{\alpha} = (n_1 + 2n_2 + 2)$; note that we have only one agent, so the vector has length one. Let I' denote the created instance $I' = (G_1 \uplus G_2^{(1)} \uplus G_2^{(2)}, \boldsymbol{\alpha})$.

The following claim builds a connection between the satisfiability of a formula and the size of a minimum dominating set.

Claim 3.35. *If φ_1 (resp. φ_2) is satisfiable, then graph G_1 (resp. $G_2^{(1)}$ and $G_2^{(2)}$ each) has a minimum dominating set of size n_1 (resp. n_2). If φ_1 (resp. φ_2) is unsatisfiable, then every minimum dominating set of G_1 (resp. $G_2^{(1)}$ and $G_2^{(2)}$) has size $n_1 + 1$ (resp. $n_2 + 1$).*

Proof. We show the statements only for G_1; the reasoning for $G_2^{(1)}$ and $G_2^{(2)}$ is analogous: The gadgets used in the constructions are identical. First, observe that for every variable $x \in X_1$, the dummy d_x is only adjacent to v_x and $v_{\overline{x}}$. Since there are exactly n_1 such variables, every dominating set of G_1 has to have size at least n_1.

For the first statement, assume that φ_1 is satisfiable, and let $\sigma \colon X_1 \to \{\mathsf{T}, \mathsf{F}\}$ a satisfying assignment for φ_1. We construct a dominating set V' of size n_1 as follows, which by our previous reasoning is a minimum-size dominating set: For each variable $x \in X_1$, if $\sigma(x) = \mathsf{T}$, then add v_x to V'; otherwise add $v_{\overline{x}}$ to V'. Clearly $|V'| = n_1$. It is also straightforward to verify that V' is a dominating set: For each variable $x \in X_1$, all three corresponding vertices are dominated. For each clause $C \in \varphi_1$, at least one literal in it is set to T, so by our construction the corresponding literal vertex is added to V' and dominates v_C. The special vertex v^* is dominated by any vertex in V'.

Now we show the second statement. We first note that G_1 has a dominating set of size $n_1 + 1$, since all dummies and the special vertex together form a dominating set. Now, to show that this is indeed minimum, we show that if G_1 has a dominating set of size n_1, then the formula φ_1 is satisfiable. Assume that V' is a dominating set for G_1 and of size n_1. As we established previously, the set V' must contain for every $x \in X_1$ one of the vertices $v_x, v_{\overline{x}}, d_x$. Since $|V'| = n_1$, it must contain *precisely* one of these vertices for every $x \in X_1$, and no other vertices. Construct the following assignment $\sigma \colon X_1 \to \{\mathsf{T}, \mathsf{F}\}$ with $\sigma(x) = \mathsf{T}$ if $v_x \in V'$, and $\sigma(x) = \mathsf{F}$ otherwise. We claim that σ satisfies φ_1. Consider an arbitrary clause $C \in \varphi_1$. Since V' is a dominating set and contains only vertices that correspond to

the variables, v_C must be dominated by a literal vertex $v_\ell \in V'$ such that $\ell \in C$. Clearly we have that $\ell \in \{x, \overline{x}\}$ for some variable x. If $\ell = x$, then by construction, variable x is set true, and hence satisfies C. Otherwise $\ell = \overline{x}$, meaning that $v_x \notin V'$. Then, variable x is set to false, and hence also satisfies C. (end of the proof of Claim 3.35) $\diamond$

Using Claim 3.35, we can show the correctness, i.e., instance (φ_1, φ_2) is a yes-instance of SAT-UNSAT if and only if I is a yes-instance of PCOG-MinDS-CV.

For the "only if" direction, assume that (φ_1, φ_2) is a yes-instance, i.e., φ_1 is satisfiable but φ_2 not. By Claim 3.35, the graph G_1 has a minimum dominating set of size n_1 while every minimum dominating set of $G_2^{(1)}$ (resp. $G_2^{(2)}$) has size $n_2 + 1$. This means that every minimum dominating set of the entire graph has size $n_1 + 2(n_2 + 1)$, since the graphs are disjoint. By definition, allocation $\boldsymbol{\alpha} = (n_1 + 2n_2 + 2)$ is a preimputation and also core-stable, witnessing that I' is a yes-instance.

For the "if" direction, assume (φ_1, φ_2) is a no-instance. Then, formula φ_1 is not satisfiable, or φ_2 is satisfiable. If φ_1 is not satisfiable, then by the first statement of Claim 3.35, the size of a minimum dominating set for G_1 is $n_1 + 1$. Again by Claim 3.35, every minimum dominating set of the entire graph $G_1 \uplus G_2^{(1)} \uplus G_2^{(2)}$ has size either $n_1 + 2n_2 + 3$ (if φ_2 is not satisfiable) or $n_1 + 2n_2 + 1$ (if φ_2 is satisfiable). This implies that $\boldsymbol{\alpha}$ is not a preimputation, and hence not core-stable.

If φ_1 is satisfiable, then φ_2 must be satisfiable in order to be a no-instance. By Claim 3.35, every minimum dominating set of the entire graph $G_1 \uplus G_2^{(1)} \uplus G_2^{(2)}$ has size $n_1 + 2n_2$, again implying that $\boldsymbol{\alpha}$ is not a preimputation, and hence not core-stable.

Since in each case, we conclude that $\boldsymbol{\alpha}$ is not core-stable, instance (φ_1, φ_2) is a no-instance as well.

DP-hardness when $c_{\max} = 4$. We now show DP-hardness when each agent owns at most 4 vertices. We achieve this by providing a polynomial-time reduction from the SAT-UNSAT problem.

Let (φ_1, φ_2) be an instance of SAT-UNSAT with variable sets X_1 and X_2, respectively. We create an instance $I' = (G = (V_1 \cup \cdots \cup V_{\hat{n}}, E), \boldsymbol{\alpha})$ of PCOG-MinDS-CV with $c_{\max} = 4$. The graph G consists of two disjoint subgraphs: G_1 and G_2, where G_1 corresponds to φ_1 and G_2 to φ_2. We note that each subgraph is a slight modification of the standard reduction from 3SAT to Dominating Set.

Graph G_1. Assume that the clauses are numbered $C_1, \ldots, C_{m_1}$. We create the following agents and their vertices for G_1:

- For every variable $x \in X_1$, create a variable agent a_x, who owns a positive literal vertex v_x, a negative literal vertex $v_{\overline{x}}$, and a dummy d_x.

- For every clause $C_j \in \varphi_1$, we add a clause agent a_{C_j} who owns the vertices $\{v^z_{C_j} \mid z \in [4]\}$.
- Create a special agent a_* who owns a special vertex v^*.

We create the following edges for G_1:

- The three vertices corresponding to the same variable form a triangle: For every $x \in X_1$, add the edges $\{v_x, v_{\overline{x}}\}$, $\{v_x, d_x\}$, and $\{v_{\overline{x}}, d_x\}$.
- For every clause $C_j \in \varphi_1$, add the edges $\{v^1_{C_j}, v^2_{C_j}\}$, $\{v^2_{C_j}, v^3_{C_j}\}$, $\{v^2_{C_j}, v^4_{C_j}\}$, $\{v^3_{C_j}, v^4_{C_{j+1}}\}$, where $m_1 + 1 = 1$.
- For every clause $C_j \in \varphi_1$, add the edge $\{v_\ell, v^1_{C_j}\}$ for every literal ℓ in the clause C_j.
- For the clause $C_1 \in \varphi_1$, add the edge $\{v^4_{C_1}, v^*\}$.

Graph G_2. This graph is constructed in the same way as G_1, except that φ_2 and X_2 are the underlying CNF formula and variable set.

Let N_1 be the set of agents constructed when constructing G_1 and N_2 the set of agents constructed when constructing G_2, respectively.

We construct an allocation $\boldsymbol{\alpha}$ as follows:

- For every $z \in [2]$:
  - In the set N_z, for every $x \in X_z$, we allocate a_x cost 1.
  - In the set N_z, for every $C_j \in \varphi_z$, we allocate a_{C_j} cost 1.
- In the set N_1, we allocate a_* cost 0.
- In the set N_2, we allocate a_* cost 1.

For the sake of brevity, we define the number of clauses in the formula as $m_1 \coloneqq |\varphi_1|$ and $m_2 \coloneqq |\varphi_2|$, and we define $n_1 \coloneqq |X_1|$ and $n_2 \coloneqq |X_2|$.

We first show the following two claims, which establish a connection between our construction and the SAT-formulas.

Claim 3.36. *If φ_1 (resp. φ_2) is satisfiable, then G_1 (resp. G_2) has a minimum dominating set of size $n_1 + m_1$ (resp. $n_2 + m_2$).*

Proof. We show the statement for G_1; the reasoning for G_2 is analogous. First, observe that for every variable $x \in X_1$ the dummy d_x is only adjacent to v_x and $v_{\overline{x}}$. Therefore, we need to add one of these three vertices to every dominating set. Similarly, for every clause $C_j \in \varphi_1$, the vertex $v^2_{C_j}$ can be dominated only by vertices in $\{v^z_{C_j} \mid z \in [4]\}$, and thus at least one of them needs to be in every dominating set. Therefore, any dominating set has size at least $n_1 + m_1$.

Assume that φ_1 is satisfiable, and let $\sigma\colon X_1 \to \{\mathsf{T}, \mathsf{F}\}$ be a satisfying assignment of φ_1. We construct a dominating set V' of G_1 of size $n_1 + m_1$. For every variable $x \in X_1$, if σ assigns x to true, we add v_x to V', otherwise we add $v_{\overline{x}}$. For every clause $C_j \in \varphi_1$, we add the vertex $v^4_{C_j}$ to V'. Clearly $|V'| = n_1 + m_1$.

For every $x \in X_1$, the vertices $v_x, v_{\overline{x}}$, and d_x are dominated. Also, for every $C_j \in \varphi_1$, the vertex $v^2_{C_j}$ is dominated by $v^4_{C_j}$, the vertex $v^3_{C_j}$ by $v^4_{C_{j+1}}$ (where $m_z + 1 = 1$), and $v^4_{C_j} \in V'$. Moreover, the vertex v^* is dominated by $v^4_{C_j}$.

It remains to show that for every $C_j \in \varphi_1$, the vertex $v^1_{C_j}$ is dominated. Since σ is a satisfying assignment, there must be a literal ℓ such that $\{v^1_{C_j}, v_\ell\} \in E(G_1)$ and $v_\ell \in V'$. Thus V' is a dominating set of G_1 of size $n_1 + m_1$. (end of the proof of Claim 3.36) $\diamond$

Claim 3.37. *Let $S_X \subseteq \{a_x \mid x \in X_1\}$ (resp. $\{a_x \mid x \in X_2\}$) and let $S_\mathcal{C} \subseteq \{a_{C_j} \mid C_j \in \varphi_1\}$ (resp. $\{a_{C_j} \mid C_j \in \varphi_2\}$). If the subgraph of G_1 (resp. G_2) induced by the vertices of $S_X \cup S_\mathcal{C} \cup \{a_*\}$ admits a dominating set of size at most $|S_X| + |S_\mathcal{C}|$, then φ_1 (resp. φ_2) admits a satisfying assignment. In particular, if G_1 (resp. G_2) admits a dominating set of size $n_1 + m_1$ (resp. $n_2 + m_2$), then φ_1 (resp. φ_2) admits a satisfying assignment.*

Proof. We show the statement for G_1; the reasoning for G_2 is analogous. Observe that for every $a_x \in S_X$, the vertex d_x must be dominated by one of the vertices from $\{d_x, v_x, v_{\overline{x}}\}$. The vertices v_x and $v_{\overline{x}}$ may also dominate vertices in $\{v^1_{C_j} \mid a_{C_j} \in S_\mathcal{C}\}$, but no other vertices belonging to the clause agents. Thus we can use at most $|S_\mathcal{C}|$ many vertices to dominate the vertices in $\{v^z_{C_j} \mid a_{C_j} \in S_\mathcal{C}, z \in \{2,3,4\}\} \cup \{v^*\}$. We will now show that every clause agent in N_1 must be in $S_\mathcal{C}$. Assume towards a contradiction that this is not the case. Consider the subgraph $G^C \coloneqq G_1[\{v^z_{C_j} \mid C_j \in S_\mathcal{C}, z \in \{2,3,4\}\} \cup \{v^*\}]$. Since not every clause agent in N_1 is in $S_\mathcal{C}$, every connected component of this subgraph must be either (1) a path of length 3ℓ for some $\ell \in \mathbb{N}$ or (2) the component that contains v^*. As $G_1[\{v^z_{C_j} \mid z \in \{2,3,4\}\}]$ is connected for every $a_{C_j} \in S_\mathcal{C}$, every clause agent in $S_\mathcal{C}$ owns vertices on exactly one connected component.

Every connected component in (1) must contain vertices from k clause agents for some $k \in \mathbb{N}$. Moreover, a minimum dominating set on such a component is of size k, because one vertex can be used to dominate at most two other vertices on a path. Since the components in (1) require at least one vertex for every clause agent in it, the component in (2) must also be dominated by the number of vertices that corresponds to the number of clause agents in it. If (2) contains only $\{v^*\}$, we must dominate it with zero vertices,

which is impossible. Therefore, the component in (2) must contain at least the vertices in $\{v^2_{C_j}, v^3_{C_j}, v^4_{C_j}, v^*\}$. Without v^*, this component contains $3k$ vertices, where k is the number of clause agents whose vertices are a part of the connected component. As previously, a minimum dominating set on such a component is of size k, because one vertex can be used to dominate at most two other vertices. Moreover, this set is unique: It consists of the agents $v^2_{C_j}$ for every clause C_j whose corresponding vertices are a part of this component. However, as this set does not contain $v^4_{C_1}$, it is not a minimum dominating set of the whole component. Thus, we need at least $k+1$ vertices to cover the whole component. But this implies that we need $|S_\mathcal{C}|+1$ many vertices to dominate all the vertices in $\{v^z_{C_j} \mid a_{C_j} \in S_\mathcal{C}, z \in \{2,3,4\}\} \cup \{v^*\}$ and thus $|S_\mathcal{C}| + |S_X| + 1$ many vertices to dominate $\bigcup_{i \in S_\mathcal{C} \cup S_X \cup \{a_*\}} V_i$, a contradiction.

Thus, we must have that every clause agent in N_1 is in $S_\mathcal{C}$. Recall that we can use at most m_1 many vertices to dominate the vertices in $\{v^z_{C_j} \mid C_j \in \varphi_1, z \in \{2,3,4\}\} \cup \{v^*\}$. Therefore for every $C_j \in \varphi_1$, we need exactly one vertex from $\{v^2_{C_j}, v^3_{C_j}, v^4_{C_j}\}$ in the dominating set, as no other vertex dominates $v^2_{C_j}$. Because v^* must be dominated, we must have that $v^4_{C_1}$ is in the dominating set V'. Because $v^3_{C_1}$ must be dominated, we must have that $v^4_{C_2} \in V'$. By repeating this argument we obtain that for every $C_j \in \varphi_1$, $v^4_{C_j} \in V'$ and $v^2_{C_j}, v^3_{C_j} \notin V'$; moreover, vertex $v^1_{C_j}$ is not dominated by $v^2_{C_j}$ and must be dominated by some vertex belonging to a variable agent.

Consider assignment σ that assigns a variable $x \in X_1$ to true if $v_x \in V'$ and false otherwise. Because we can have at most one vertex from every variable agent in V'—otherwise V' would be too large—we have that $v_{\overline{x}} \in V'$ implies that σ assigns x to false. Assignment σ must be a satisfying assignment: As we established previously, for every $C_j \in \varphi_1$ there must be a literal $\ell \in C_j$ such that $v_\ell \in V'$, meaning that the literal ℓ is true under σ. (end of the proof of Claim 3.37) $\diamond$

Using Claims 3.36 and 3.37, we can show correctness, i.e., instance (φ_1, φ_2) is a yes-instance of SAT-UNSAT if and only if I' is core-stable.

For the "only if" direction, assume (φ_1, φ_2) is a yes-instance, i.e., formula φ_1 is satisfiable but φ_2 not.

Let us start by showing the following claim that restricts the blocking coalitions we must consider:

Claim 3.38. *No set of agents $S \subseteq \mathsf{N}_1 \setminus \{a_*\}$ (resp. $\mathsf{N}_2 \setminus \{a_*\}$) may block.*

Proof. We show the statement for N_1; the reasoning for N_2 is analogous. Let $S \subseteq \mathsf{N}_1 \setminus \{a_*\}$. Let S_X be the set of variable agents in S and $S_\mathcal{C}$ be the set of clause agents in S. We observe that by construction $\sum_{i \in S} \boldsymbol{\alpha}_i = |S_X| + |S_\mathcal{C}|$. Moreover, we show that $c(S) \geq |S_X| + |S_\mathcal{C}|$, i.e., the minimum dominating set of $G_1[\bigcup_{i \in S} V_i]$ is at least size $|S_X| + |S_\mathcal{C}|$. Let V' be a minimum dominating set of $G_1[\bigcup_{i \in S} V_i]$. For every variable agent $a_x \in S_X$, there must

be a vertex that dominates the dummy vertex d_x. By construction, this must be one of the three vertices owned by a_x, as no other vertices can dominate d_x. For every clause agent $a_{C_j} \in S_{\mathcal{C}}$, vertex $v^2_{C_j}$ must be dominated by one of the vertices in $\{v^z_{C_j} \mid z \in [4]\}$. As none of the above enumerated sets overlap, we must have that $|V'| \geq |S_X| + |S_{\mathcal{C}}|$. Thus $\sum_{i \in S} \boldsymbol{\alpha}_i = |S_X| + |S_{\mathcal{C}}| \leq c(S)$, as required. (end of the proof of Claim 3.38) $\diamond$

Since graphs G_1 and G_2 are not connected to each other, by Observation 3.13 it is sufficient for us to show that $\boldsymbol{\alpha}$ restricted to each of these graphs is core-stable. We have four possible cases:

Case 1: A subset $S \subseteq \mathsf{N}_1$ blocks $\boldsymbol{\alpha}$ restricted to N_1. By Claim 3.38, we have that $a_* \in S$ and $\sum_{i \in S \setminus \{a_*\}} \boldsymbol{\alpha}_i \leq c(S \setminus \{a_*\})$. Since $\boldsymbol{\alpha}_{a_*} = 0$, it holds that $\sum_{i \in S \setminus \{a_*\}} \boldsymbol{\alpha}_i = \sum_{i \in S} \boldsymbol{\alpha}_i$. Additionally, since v^* is adjacent only to $v^4_{C_1}$, we have that $c(S \setminus \{a_*\}) \leq c(S)$. Thus we obtain that $\sum_{i \in S} \boldsymbol{\alpha}_i = \sum_{i \in S \setminus \{a_*\}} \boldsymbol{\alpha}_i \leq c(S \setminus \{a_*\}) \leq c(S)$, a contradiction to S blocking.

Case 2: We have that $\boldsymbol{\alpha}$ restricted to N_1 is not a preimputation. Observe that $\sum_{i \in \mathsf{N}_1} \boldsymbol{\alpha}_i = n_1 + m_1$. As the previous case implies that $\sum_{i \in \mathsf{N}_1} \boldsymbol{\alpha}_i \leq c(\mathsf{N}_1)$, we must have that $n_1 + m_1 < c(\mathsf{N}_1)$, i.e., a minimum dominating set of G_1 is strictly larger than $n_1 + m_1$. However, as φ_1 admits a satisfying assignment, the graph G_1 admits a dominating set of size $n_1 + m_1$ by Claim 3.36, a contradiction.

Case 3: A subset $S \subseteq \mathsf{N}_2$ blocks $\boldsymbol{\alpha}$ restricted to N_2. By Claim 3.38, we have that $a_* \in S$. Let S_X be the set of variable agents in S and $S_{\mathcal{C}}$ be the set of clause agents in S. We have that $\sum_{i \in S} \boldsymbol{\alpha}_i = |S_X| + |S_{\mathcal{C}}| + 1$. Thus we must have that $G_2[\bigcup_{i \in S} V_i]$ admits a dominating set V' of size at most $|S_X| + |S_{\mathcal{C}}|$. By Claim 3.37, the formula φ_2 is then satisfiable. This, however, contradicts that (φ_1, φ_2) is a yes-instance of SAT-UNSAT.

Case 4: We have that $\boldsymbol{\alpha}$ restricted to N_2 is not a preimputation. The previous case establishes that $\sum_{i \in \mathsf{N}_2} \boldsymbol{\alpha}_i \leq c(\mathsf{N}_2)$, as otherwise N_2 would block. Thus we must have that $c(\mathsf{N}_2) > \sum_{i \in \mathsf{N}_2} \boldsymbol{\alpha}_i = n_2 + m_2 + 1$, i.e., a minimum dominating set of G_2 is strictly larger than $n_2 + m_2 + 1$. Towards a contradiction, we construct a dominating set V' of G_2 of size $n_2 + m_2 + 1$. Let $V' \coloneqq \{v_x \mid x \in X_2\} \cup \{v^2_{C_j} \mid C_j \in \varphi_2\} \cup \{v^*\}$. Clearly $|V'| = n_2 + m_2 + 1$. For every variable $x \in X_2$, the vertices $v_{\overline{x}}$ and d_x are dominated by v_x. For every clause $C \in \varphi_2$, the vertices $v^1_{C_j}, v^3_{C_j}$, and $v^4_{C_j}$ are dominated by $v^4_{C_j}$. Thus, every vertex is either in V' or dominated by a vertex in V'.

This concludes the proof of the "only if" direction.

For the "if" direction, assume I' is a yes-instance of PCOG-MinDS-CV, i.e., allocation $\boldsymbol{\alpha}$ is core-stable. Because G_1 and G_2 are not connected, we must have that $\boldsymbol{\alpha}$ is core-stable when restricted to both of these graphs by Observation 3.13.

We first show that φ_1 admits a satisfying assignment. Since $\sum_{i \in \mathsf{N}_1} \boldsymbol{\alpha}_i = n_1 + m_1$, the graph G_1 must admit a dominating set V' of size $n_1 + m_1$. By Claim 3.37, this implies that φ_1 admits a satisfying assignment.

Next, we show that φ_2 cannot admit a satisfying assignment. Since $\sum_{i \in \mathsf{N}_2} \boldsymbol{\alpha}_i = n_2 + m_2 + 1$, the graph G_2 cannot admit a dominating set V' of size $n_2 + m_2$. By the contrapositive of Claim 3.36, the formula φ_2 cannot admit a satisfying assignment. Thus (φ_1, φ_2) is a yes-instance of SAT-UNSAT. □

3.4.2 Core-Stability Existence

Through a technique analogous to the one for Theorem 3.21, we also obtain Θ_2^{P}-containment for dominating set.

Theorem 3.39. *PCOG-MinDS-CE is contained in* Θ_2^P.

Proof. The proof is highly analogous to the proof of Theorem 3.21. Instead of showing coNP-containment of Hinted-VC, we show the coNP-containment of a hinted version of PCOG-MinDS-CE:

Hinted-PCOG-MinDS-CE
Input: An instance $G = (V = V_1 \cup \cdots \cup V_{\hat{n}}, E)$ and k^* which is the size of a minimum dominating set of G.
Question: Is G a yes-instance of PCOG-MinDS-CE?

The proof is analogous to Claim 3.22 on page 63, with dominating sets in place of vertex covers: For a no-instance (G, k^*), let $\mathcal{T} \coloneqq ((\boldsymbol{\alpha}^1, \omega^1), \ldots, (\boldsymbol{\alpha}^r, \omega^r))$ be the transcript of one execution of the ellipsoid method on (G, k^*), where $\boldsymbol{\alpha}^j$ is the j-th queried allocation and ω^j is the answer of the separation oracle. We obtain a witness $\mathcal{W}$ by replacing every violated constraint ω^j with a pair (S_j, V_j'), where $S_j \subseteq \mathsf{N}$ is a coalition and V_j' is a dominating set of $G[\bigcup_{i \in S_j} V_i]$ with $|V_j'| < \sum_{i \in S_j} \boldsymbol{\alpha}_i^j$, for every $j \in [r]$. As k^* is given, we once again assume that all queried allocations are preimputations. With the same argument, as in Claim 3.22, $\mathcal{W}$ has size polynomial in the input. To verify $\mathcal{W}$, we re-simulate the deterministic ellipsoid method on (G, k^*). For each queried allocation $\boldsymbol{\alpha}^j$, we use the input value k^* to check that $\boldsymbol{\alpha}^j$ is a preimputation, and verify in polynomial time that V_j' is a dominating set of $G[\bigcup_{i \in S_j} V_i]$ with $|V_j'| < \sum_{i \in S_j} \boldsymbol{\alpha}_i^j$, which certifies that S_j blocks $\boldsymbol{\alpha}^j$. The verifier accepts if and only if the simulated ellipsoid method concludes that the corresponding polytope is empty, and it runs in polynomial time. The rest of the proof proceeds as in Theorem 3.21: We find the size k^* of a minimum dominating set of G through $\log(|V|)$ calls to an NP-oracle solving Dominating Set, and then conclude with one further NP-oracle call to the co-problem of Hinted-PCOG-MinDS-CE. □

We complement the Θ_2^{P}-containment result above by showing Θ_2^{P}-hardness, thereby showing Θ_2^{P}-completeness. The proof is similar to the one for Theorem 3.23.

Theorem 3.40. *PCOG-MinDS-CE is* Θ_2^P*-hard even when* $\hat{n} = 4$.

Proof. We first show the Θ_2^{P}-hardness of the following problem.

Dominating Set Membership
Input: A graph $G = (V, E)$ and a vertex $v' \in V$.
Question: Does G admit a *minimum* dominating set V' such that $v' \in V'$?

Claim 3.41. *Dominating Set Membership is Θ_2^P-hard.*

Proof. To show hardness, we reduce from Vertex Cover Membership. We utilize the standard reduction from Vertex Cover to Dominating Set. Let $I = (G = (V, E), v')$ be an instance of Vertex Cover Membership. We now create the following instance $I' = (G' = (U, E'), u')$ for Dominating Set Membership: Let us first construct the vertices.

- For each edge $e \in E$, we add a vertex u_e. Vertices added in this way are referred to as edge-vertices.
- For each vertex $v \in V$, we add a vertex u_v. Vertices added in this way are referred to as vertex-vertices.

Next, we construct the edges.

- For each $v, w \in V, v \neq w$, we add the edge $\{u_v, u_w\}$
- For each $e \in E, v \in e$, we add the edge $\{u_v, u_e\}$.

Finally, we set $u' \coloneqq u_{v'}$.

We start by noting that there exists a minimum dominating set of G' consisting only of vertex-vertices: Given a dominating set U' of U containing an edge vertex $u_e, e \in e$, we can construct a dominating set U'' with $|U''| \leq |U'|$ by replacing u_e with u_v such that $v \in e$, since u_v dominates u_e and every vertex adjacent to u_e.

Given a vertex cover V' of G, we can verify that $U' \coloneqq \{u_v \mid v \in V'\}$ is a dominating set of G' such that $|U'| = |V'|$, and $v' \in V'$ if and only if $u_{v'} \in U'$. The fact that it is a dominating set follows, since each edge-vertex is dominated by a vertex-vertex, since V' is a vertex cover, and each vertex-vertex is dominated by every other vertex-vertex. Similarly, given a dominating set $U'' \subseteq U$ such that U'' only contains vertex-vertices, we can verify that $V'' \coloneqq \{v \in V \mid u_v \in U''\}$ is a vertex cover of G such that $|U''| = |V''|$ and $v' \in V''$ if and only if $u_{v'} \in U''$. This follows as each edge-vertex must be dominated by a vertex-vertex. Therefore, it follows directly that v' is in a minimum vertex cover in G if and only if $u_{v'}$ is in a minimum dominating set of G', thereby showing the statement.

(end of the proof of Claim 3.41) $\diamond$

We can now reduce from Dominating Set Membership to PCOG-MinDS-CE. Let $I = (G = (V, E), v')$ be an instance of Dominating Set Membership. We now generate an instance $I' = ([4], G' = (V_1 \cup V_2 \cup V_3 \cup V_4, E'))$ of PCOG-MinDS-CE that contains four agents. We describe the instance I' by describing the sub-graph induced by each agent's vertices first, and then describing the edges connecting these sub-graphs.

- $G'[V_1], G'[V_2], G'[V_3]$ are all cliques of size 3, i.e., triangles. We name the vertices in the following way: $V_1 = \{v_{1,1}, v_{1,2}, v_{1,3}\}$, $V_2 = \{v_{2,1}, v_{2,2}, v_{2,3}\}$, $V_3 = \{v_{3,1}, v_{3,2}, v_{3,3}\}$.
- $G[V_4] = (V, E)$, i.e., the agent has as its induced subgraph exactly the graph given in I. The vertices will keep the naming in I. In particular, vertex v' will keep its name.
- We add the following edges connecting the sub-graphs $\{v_{i,j}, v_{j,k}\}$ for $i, j, k \in [3]$ and $i \neq j$. In other words, for each triangle, there exists a vertex for each other triangle that dominates all vertices in that triangle. Finally, we add edges $\{v', v_{1,i}\}$ for $i \in [3]$.

We claim that I is a yes-instance of Dominating Set Membership if and only if I' is a yes-instance of PCOG-MinDS-CE.

For the "only if" direction, assume I is a yes-instance of Dominating Set Membership. Let k^* be the minimum dominating set size of G. We claim that $\boldsymbol{\alpha} = (k^*, 0, 0, 1)$ is an allocation in the core. We start by noting that the minimum dominating set size of the graph must be of size at least $k^* + 1$, as it is not possible to dominate the vertices in $V_1 \cup V_2 \cup V_3$ using any single vertex. As only v' in V_4 can be used to dominate vertices $V_1 \cup V_2 \cup V_3$, namely V_1, it follows that the minimum dominating set size is at least $k^* + 1$.

Let V' be a minimum dominating set of size k^* containing v'. The set $V' \cup \{v_{3,2}\}$ dominates all vertices in the graph, as vertices in V_4 are dominated by V', vertices in V_1 are dominated by v', and vertices in V_2 and V_3 are dominated by $v_{3,2}$. We now show that the allocation $(k^*, 0, 0, 1)$ is not blocked. Suppose agent 4 is part of an inclusionwise minimal blocking coalition. Then agent 1 must be part of the blocking coalition as well, as the other agents' vertex sets do not have vertices that are adjacent to vertices in V_4. However, agents 1 and 4 have a total cost of k^*, and adding any additional agent would increase the cost of the blocking coalition to $k^* + 1 = \sum_{i \in [4]} \boldsymbol{\alpha}_i$. For every subset $S \subseteq [3]$, it holds that $\sum_{i \in S} \boldsymbol{\alpha}_i \leq 1 \leq c(S)$, contradicting S blocking.

For the "if" direction, assume I' is a yes-instance of PCOG-MinDS-CE. Let $\boldsymbol{\alpha}$ be a core-stable allocation. No pair of agents in [3] can have a total cost of more than 1, as the minimum dominating set size for each pair is exactly 1. From this, it follows that $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 < 2$. Let k^* be the minimum dominating set size of G'. Thereby, it follows that $\boldsymbol{\alpha}_4 > k^* - 2$, and therefore the size of a minimum dominating set of $G[V_4]$ is lower-bounded by $k^* - 1$. This implies that G must admit a minimum-size dominating set containing v', as the only vertex that can be used to dominate vertices in $V_1 \cup V_2 \cup V_3$

is v', and it is not possible to cover the vertices in $V_1 \cup V_2 \cup V_3$ using only a single vertex from $V_1 \cup V_2 \cup\ V_3$. □

Next, we show that the problem remains Θ_2^{P}-hard even when $c_{\max} = 9$, by reducing from PCOG-MinVC-CE.

Theorem 3.42. *PCOG-MinDS-CE is Θ_2^P-hard even when $c_{max} = 9$.*

Proof. We reduce from PCOG-MinVC-CE, which by Theorem 3.23 is Θ_2^{P}-hard even if $c_{\max} = 9$. Let $I = G = (V, E = E_1 \cup \cdots \cup E_{\hat{n}})$ be an instance of PCOG-MinVC-CE. We construct a reduced instance $I' = G' = (U, E')$ of PCOG-MinDS-CE that is inspired by the standard reduction from Vertex Cover to Dominating Set. The vertices are as follows:

- For every $v \in V$, we construct a vertex-vertex u_v in G'.
- For every $e \in E$, we construct an edge-vertex $u_e \in G'$.

Let $U_V \coloneqq \{u_v \mid v \in V\}$ and $U_E \coloneqq \{u_e \mid e \in E\}$. The edges are as follows:

- For every $v, v' \in V, v \neq v'$, we add the edge $\{u_v, u_{v'}\}$.
- For every $e = \{v, v'\} \in E$, we add the edges $\{u_v, u_e\}$ and $\{u_{v'}, u_e\}$.

The agent set N' is as follows:

- For every $i \in [\hat{n}]$, we create an agent i' that owns the vertices $U_{i'} \coloneqq \{u_e \mid e \in E_i\}$. Clearly, i' owns the same number of vertices as the number of edges owned by i in I.
- For every $v \in V$, we create an agent i_v that owns the vertex set $U_v \coloneqq \{u_v\}$.

Let $\mathsf{N}_V \coloneqq \{i_v \mid v \in V\}$ and $U_V \coloneqq \{u_v \mid v \in V\}$. Observe that the maximum organization size of I' is equal to the maximum organization size of I, and is thus a constant.

Claim 3.43. *Let $\emptyset \subsetneq U' \subseteq U$ and let V' be a dominating set of $G[U']$. Then there is a dominating set V'' of $G[U' \cup U_V]$ such that $|V''| \leq |V'|$ and $V'' \subseteq U_V$.*

Proof. We construct the dominating set V'' as follows: For every $e \in E$ such that $u_e \in V'$, we remove u_e from V' and replace it with a vertex u_v such that $v \in e$, unless such u_v is already in V'. Since u_v dominates every vertex that u_e dominates, if V' is a dominating set of U', so is V''. Clearly $|V''| \leq |V'|$ and $V'' \subseteq U_V$. Because V' is non-empty, the set V'' must contain at least one vertex from U_V, and thus every vertex in U_V is dominated. (end of the proof of Claim 3.43) ⋄

Claim 3.44. *Let $\tilde{E} \subseteq E$ and $k^* \in [|V|]$. Then $G[\tilde{E}]$ admits a vertex cover of size at most k^* if and only if $G'[\{u_e \mid e \in \tilde{E}\} \cup U_V]$ admits a dominating set of size at most k^*.*

Proof. Assume $G[\tilde{E}]$ admits a vertex cover V' of size at most k^*. Then we construct a dominating set $V'' \coloneqq \{u_v \mid v \in V'\}$ of $G'[\{u_e \mid e \in \tilde{E}\} \cup U_V]$. Clearly $|V''| = |V'|$. Every vertex in $\{u_e \mid e \in \tilde{E}\} \cup U_V$ is also dominated: Because V' is non-empty, the set V'' must contain at least one vertex from U_V, and thus every vertex in U_V is dominated. Because V' is a vertex cover, for every $e \in \tilde{E}$ there is $v \in e \cap V'$. Therefore, $u_v \in V''$ and the vertex u_e is dominated, as required.

Now assume $G'[\{u_e \mid e \in \tilde{E}\} \cup U_V]$ admits a dominating set V'' of size at most k^*. By Claim 3.43, we can assume that $V'' \subseteq U_V$. Now, let us construct a vertex cover $V' \coloneqq \{v \in V \mid u_v \in V''\}$. Clearly $|V'| = |V''| \leq k^*$. Moreover, for every $e \in \tilde{E}$ there must be a vertex $u_v \in V''$ that dominates u_e. By construction, the corresponding vertex $v \in V$ covers the edge e, and we have that $v \in V'$. Thus, V' is a vertex cover. (end of the proof of Claim 3.44) $\diamond$

We now show the correctness of our reduction.

For the "only if" direction, assume that PCOG-MinVC-CE-instance I admits a core-stable allocation $\boldsymbol{\alpha}$. We construct a core-stable allocation $\boldsymbol{\alpha}'$ of I' as follows: For every $i \in [\hat{n}]$, $\boldsymbol{\alpha}'_{i'} \coloneqq \boldsymbol{\alpha}_i$ and for every $v \in V$, $\boldsymbol{\alpha}'_{i_v} \coloneqq 0$.

We first show that G' admits a dominating set of size at most $\sum_{a \in \mathsf{N}'} \boldsymbol{\alpha}'_a = \sum_{i \in [\hat{n}]} \boldsymbol{\alpha}_i$, and thus $\sum_{a \in \mathsf{N}'} \boldsymbol{\alpha}'_a \geq c(\mathsf{N})$. Let V' be a minimum vertex cover of G. Because $\boldsymbol{\alpha}$ is an allocation, we have that $\sum_{i \in [\hat{n}]} \boldsymbol{\alpha}_i = c(\mathsf{N}) = |V'|$. By Claim 3.44, the graph G' admits a dominating set of size at most $|V'|$. Thus $c(\mathsf{N}') \leq |V'| = \sum_{a \in \mathsf{N}'} \boldsymbol{\alpha}'_a$, as required.

Next we show that for every $S' \subseteq \mathsf{N}'$, it holds that $\sum_{a \in S'} \boldsymbol{\alpha}'_a \leq c(S')$. Assume, towards a contradiction, that there is $S' \subseteq \mathsf{N}'$ such that $\sum_{a \in S'} \boldsymbol{\alpha}'_a < c(S')$.

Let V'' be a minimum dominating set of $G'[\bigcup_{a \in S'} U_a]$. By Claim 3.43, there is a dominating set $\hat{V}''$ of $S' \cup \mathsf{N}_V$ such that $|\hat{V}''| \leq |V''|$. Let $S \coloneqq \{i \in \mathsf{N} \mid i' \in S'\}$. By Claim 3.44, the graph $G[S]$ admits a vertex cover V' such that $|V'| \leq |\hat{V}''|$. We have that

$$\sum_{i \in S} \boldsymbol{\alpha}_i = \sum_{a \in S'} \boldsymbol{\alpha}'_a > c(S') = |V''| \geq |\hat{V}''| \geq |V'| \geq c(S),$$

a contradiction to $\boldsymbol{\alpha}$ being core-stable.

For the "if" direction, assume I' admits a core-stable allocation $\boldsymbol{\alpha}'$. We construct an allocation $\boldsymbol{\alpha}$ of I as follows: Let $f \colon \mathsf{N}_V \to \mathsf{N}$ be an arbitrary function such that for every $v \in U_V$ it holds that $f(i_v)$ owns an edge that is incident to v. In other words, the function f assigns every vertex agent in N_V to some agent in N that has an edge that is incident to this vertex. For every $i \in \mathsf{N}$, let $\boldsymbol{\alpha}_i \coloneqq \boldsymbol{\alpha}'_{i'} + \sum_{i_v \in f^{-1}(i)} \boldsymbol{\alpha}'_{i_v}$, i.e., every agent obtains the cost of the corresponding agent from N' and the cost of any vertex agents from N_V that are assigned to it.

First we observe that $\sum_{i\in\mathsf{N}} \boldsymbol{\alpha}_i = \sum_{a\in\mathsf{N}'} \boldsymbol{\alpha}'_a \geq c(\mathsf{N}')$. Since $\boldsymbol{\alpha}'$ is an allocation, a minimum dominating set of G' is of size $c(\mathsf{N}')$. By Claim 3.44, the graph G must admit a vertex cover of size at most $c(\mathsf{N}')$, and thus $\sum_{i\in\mathsf{N}} \boldsymbol{\alpha}_i \geq c(\mathsf{N})$.

Next we show that for every $S \subseteq \mathsf{N}$, it holds that $\sum_{i\in S} \boldsymbol{\alpha}_i \leq c(S)$. Assume, towards a contradiction, that there is $S \subseteq \mathsf{N}$ such that $\sum_{i\in S} \boldsymbol{\alpha}_i > c(S)$. Let V' be a minimum vertex cover of $G[\cup_{i\in S} E_i]$. Let $S' \coloneqq \{i' \mid i \in S\}$. Let V'' be the minimum dominating set of $G'[\bigcup_{a\in S'\cup\mathsf{N}_V} U_a]$. Due to Claim 3.43, we can assume that $V'' \subseteq U_V$ and by Claim 3.44, it holds that $|V''| \leq |V'|$. We can also assume that every vertex in V'' is adjacent to a vertex in $\bigcup_{i\in S} U_{i'}$, as otherwise we could remove this vertex and obtain a smaller dominating set: Since $\bigcup_{i\in S} U_{i'}$ is non-empty and $V'' \subseteq U_V$, there is at least one vertex v' in V'' that is incident to a vertex in $\bigcup_{i\in S} U_{i'}$, and v' dominates every vertex in U_V.

Let N^*_V be the set of agents $i_v, v \in V$ such that v is incident to an edge in $\cup_{i\in S} E_i$. By construction and earlier reasoning we have that $V'' \subseteq \{u_v \mid v \in V, i_v \in \mathsf{N}^*_V\}$. Observe that by construction of $\boldsymbol{\alpha}$ and the fact that $\boldsymbol{\alpha}'_a \geq 0$ for every $a \in \mathsf{N}'$

$$\sum_{a\in S'\cup\mathsf{N}^*_V} \boldsymbol{\alpha}'_a = \sum_{i'\in S'} \boldsymbol{\alpha}'_{i'} + \sum_{a\in\mathsf{N}^*_V} \boldsymbol{\alpha}'_a \geq \sum_{i'\in S'} \boldsymbol{\alpha}'_{i'} + \sum_{i'\in S'} \sum_{v\in f^{-1}(i)} \boldsymbol{\alpha}'_{i_v} = \sum_{i\in S} \boldsymbol{\alpha}_i. \tag{3.2}$$

Since V'' is a dominating set of $G'[S'\cup\mathsf{N}_V]$ and $V'' \subseteq \mathsf{N}_V$, the set V'' is also a dominating set of $G'[S'\cup\mathsf{N}^*_V]$. We obtain that

$$c(S'\cup\mathsf{N}^*_V) \leq |V''| \leq |V'| = c(S) \tag{3.3}$$

By combining Equations (3.2) and (3.3) and $\sum_{i\in S} \boldsymbol{\alpha}_i > c(S)$, we obtain that

$$\sum_{a\in S'\cup\mathsf{N}^*_V} \boldsymbol{\alpha}'_a \geq \sum_{i\in S} \boldsymbol{\alpha}_i > c(S) \geq c(S'\cup\mathsf{N}^*_V),$$

a contradiction to the core stability of $\boldsymbol{\alpha}$'.

This concludes the "if" direction and hence the proof. □

Finally, we prove the conjecture of van Velzen [vV04] by showing that the unpartitioned variant COG-MinDS-CE is indeed NP-hard.The correctness of the reduction is based on the characterization result of Deng [Den09] for when a core-stable allocation exists, which van Velzen [vV04] explicitly shows for the MinDS case. The characterization result also implies NP-containment.

Theorem 3.45. *COG-MinDS-CE is* NP*-complete and hence, PCOG-MinDS-CE is* NP*-complete when* $c_{max} = 1$.

As already mentioned, we will use the following lemma from van Velzen [vV04]. To this end, define *fractional dominating set value* of a graph G as the objective value

of an optimal solution to the following linear program: For every $v \in V$, we have a variable $y_v \geq 0$.

$$\text{Minimize } \sum_{v \in V} y_v \text{ subject to} \qquad \text{(LP1)}$$
$$\forall\, v \in V \sum_{v' \in N_G(v)} y_{v'} + y_v \geq 1$$

Lemma 3.46 ([vV04], Theorem 4.2)**.** *An instance $G = (V, E)$ of COG-MinDS-CE is a yes-instance if and only if the fractional dominating set value of G is equal to the size of a minimum dominating set of G.*

We are ready to show Theorem 3.45.

Proof. We use the standard reduction from 3SAT to Dominating Set. Let φ be a SAT-formula over the set of n variables X. We construct the input graph $G = (V, E)$ as follows:

- For each variable $x \in X$, we add three vertices corresponding to the two literals, x and $\overline{x}$, and a dummy. For each clause $C \in \varphi$, we add the vertex v_C.
- We add an edge between v_ℓ and v_C, if the literal ℓ appears in the clause C. We add an edge between each pair in $\{v_x, v_{\overline{x}}, v_{x^d}\}$.

Since $c_{\max} = 1$ (equivalently, the constructed instance is an instance of COG-MinDS-CE), every vertex is owned by its own agent.

We obtain the following structural result:

Claim 3.47. *The fractional dominating set value of G is n.*

Proof. We first show that the fractional dominating set value of G is at least n. Consider the linear program corresponding to G as described in LP1. For the ease of presentation, for every $x \in X$, we denote the variable corresponding to v_x by y_x, the variable corresponding to $v_{\overline{x}}$ by $y_{\overline{x}}$, and the variable corresponding to v_{x^d} by y_{x^d}. Similarly, for every clause $C \in \varphi$, we denote the variable corresponding to v_C by y_C.

Observe that for every $x \in X$, the constraint of v_{x^d} states that $y_x + y_{\overline{x}} + y_{x^d} \geq 1$. Thus $\sum_{v \in V} y_v \geq \sum_{x \in X}(y_x + y_{\overline{x}} + y_{x^d}) \geq n$.

Next, we show that the linear program admits a solution of size n. For every $x \in X$, let $y_x \coloneqq \frac{1}{2}$ and $y_{\overline{x}} \coloneqq \frac{1}{2}$. We show that this satisfies the solution: For every $x \in X$, the constraint of v_{x^d} states that $y_x + y_{\overline{x}} + y_{x^d} \geq 1$. We have that $\frac{1}{2} + \frac{1}{2} \geq 1$, so the constraint is satisfied. The constraint of v_x states that $y_x + y_{\overline{x}} + y_{x^d} + \sum_{C \in \varphi, x \in C} y_C \geq 1$. Again $\frac{1}{2} + \frac{1}{2} \geq 1$. We can show that the constraint of $v_{\overline{x}}$ is satisfied analogously.

For every C, we may assume that C contains at least two literals. The constraint of v_C is $\sum_{\ell \in C} y_\ell \geq 1$. Observe that $y_\ell = \frac{1}{2}$ for every $\ell \in C$. We have that $|C|\frac{1}{2} \geq 1$, and thus the constraint is satisfied.

Thus the fractional dominating set value of G is n. (end of the proof of Claim 3.47) $\diamond$

For the "only if" direction, assume φ admits a satisfying assignment σ. By reasoning of the proof of Claim 3.35 from the proof of Theorem 3.34, graph G admits a dominating set of size n: Observe that the process of constructing G is the same as the process of constructing G_1, $G_2^{(1)}$, and $G_2^{(2)}$ from the proof of Theorem 3.34, except for the lack of a special vertex. As the dominating set constructed in the proof of Claim 3.35 does not contain a special vertex, we can obtain a dominating set of G by using this construction.

By Claim 3.47, graph G admits a dominating set of size n and thus the size of a minimum dominating set and the fractional dominating set value of G are equal. By Lemma 3.46, the instance admits a core-stable allocation.

For the "if" direction, assume that G admits a core-stable allocation. Then by Lemma 3.46 the fractional dominating set value and the size of a minimum dominating set of G are equal, and by Claim 3.47, graph G admits a dominating set V' of size n. In the proof of Claim 3.35 from the proof of Theorem 3.34, we construct a satisfying assignment σ when G_1 admits a dominating set of size n_1. Since this dominating set does not contain the special vertex, we can use this construction to construct a satisfying assignment of φ from the dominating set V'. This concludes the proof. □

3.5 Partitioned Maximum Matching Game and Partitioned Minimum Spanning Tree Game

In this section, we discuss the computational complexity of partitioned maximum matching and minimum spanning tree games. Unlike the previously studied games, here the underlying optimization problem is tractable, which is reflected by a lower complexity of CV and CE for the corresponding games.

For MinST, Faigle et al. [FKFH97] show that the unpartitioned variant COG-MinST-CV is coNP-complete. We observe that their approach for showing coNP-containment can be adapted for our partitioned variant:

Observation 3.48. *PCOG-MinST-CV is contained in* coNP*.*

Proof. Given an allocation $\boldsymbol{\alpha}$ and blocking coalition S, we can compute the minimum weight spanning tree on the subgraph induced by the vertices of the agents in S and verify whether it is less than $c(S)$. Thus, S is a polynomial-time verifiable witness for a no-instance, meaning that PCOG-MinST-CV is contained in coNP. □

Regarding the existence question, Proposition 3.12 implies that every PCOG-MinST-CE-instance is a yes-instance and a core-stable allocation can be found in polynomial time.

Next, we turn to MaxMatch. Biró et al. [BKPP19] show that PCOG-MaxMatch-CE is coNP-hard and it is polynomial-time solvable when $c_{\max} = 2$. We complement the hardness result by proving coNP-containment, which utilizes the ellipsoid method.

Proposition 3.49. *PCOG-MaxMatch-CE is contained in* coNP*.*

Proof. We show coNP-containment using the ellipsoid method; see Section 2.3.2 for more details. We show a polynomial-time verifiable certificate for a no-instance. Our proof proceeds in two parts: In part 1, we show that every no-instance of PCOG-MaxMatch-CE admits a polynomial-size witness, and in part 2, we show how to verify the witness. The argument is a simplified version of the proof of Claim 3.22 on page 63. As we can compute a maximum matching in polynomial time, we do not require the size of a maximum matching, and our separation oracle only needs to return a blocking coalition.

Part 1. We start with an informal description: First, recall that there exists an LP formulation for every TU-game (cf. Section 2.4.2). Suppose we are given a separation oracle (cf. Definition 2.39) as a blackbox algorithm. Then, by Theorem 2.40, the ellipsoid method would solve the problem in polynomial time. In the case of a no-instance, the separation oracle calls always return violated constraints, which consist of a coalition of players S and the size of a maximum matching for the graph induced by S. Informally, our witness will consist of a sequence of coalitions. These witnesses will replace our separation oracle calls, and thereby allow us to execute the ellipsoid method with all our separation oracle calls already being available in the witness. Note that we can compute a maximum matching for $G[\bigcup_{i\in S} V_i]$ in polynomial time, and therefore, we do not require a maximum-size matching as part of a witness.

Let G be a no-instance of PCOG-MaxMatch-CE. Observe that an allocation $\boldsymbol{\alpha}$ is core-stable if and only if it satisfies the core stability constraints from Definition 2.51[9]. In other words, $\boldsymbol{\alpha}$ is core-stable if and only if it belongs to the polytope

$$P \coloneqq \left\{ \boldsymbol{\alpha} \in \mathbb{R}^{\hat{n}}_{\geq 0} \;\middle|\; \sum_{i\in \mathsf{N}} \boldsymbol{\alpha}_i = \nu(\mathsf{N}) \text{ and } \sum_{i\in S} \boldsymbol{\alpha}_i \geq \nu(S)\ \forall S \subseteq \mathsf{N} \right\}, \tag{3.4}$$

where $\nu(S)$ denotes the size of a maximum matching of $G[\bigcup_{i\in S} V_i]$. Thus, G is a yes-instance of PCOG-MaxMatch-CE if and only if $P \neq \emptyset$. The polytope P is given implicitly through the graph G, and therefore, we use the separation oracle version of the ellipsoid method.

Let Q be a separation oracle for P. Given an allocation $\boldsymbol{\alpha}$, oracle Q either certifies that $\boldsymbol{\alpha} \in P$, or returns a violated constraint of P. In other words,

[9] Section 2.4 also provides an explicit LP formulation.

- if $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i \neq \nu(\mathsf{N})$, then Q returns the violated preimputation constraint $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = \nu(\mathsf{N})$; and
- otherwise, if there is a coalition $S \subseteq \mathsf{N}$ with $\nu(S) > \sum_{i \in S} \boldsymbol{\alpha}_i$, then Q returns the violated coalition constraint $\sum_{i \in S} \boldsymbol{\alpha}_i \geq \nu(S)$.

Since G does not admit a core-stable allocation, every query made by the ellipsoid method yields a violated constraint. Let

$$\mathcal{T} \coloneqq ((\boldsymbol{\alpha}^1, \omega^1), \ldots, (\boldsymbol{\alpha}^r, \omega^r))$$

denote the transcript of one execution of the ellipsoid method on G, where $\boldsymbol{\alpha}^j$ is the j-th queried allocation and ω^j is the violated constraint returned by Q. By Theorem 2.40, the ellipsoid method queries the separation oracle a polynomial number of times, as both the dimension (the number of agents in our case) and the maximum encoding length of a constraint[10] are bounded by the input size. Hence, r is polynomially bounded in $|V|$. Unlike in Claim 3.22, each ω^j is determined by its coalition S_j alone, because $\nu(S_j)$ is computable in polynomial time; no cover certificate is needed. Replacing each ω^j by the coalition S_j witnessing it yields the witness $\mathcal{W} \coloneqq (S_1, \ldots, S_r)$ of size polynomial in $|V|$.

Part 2. Assume we are given $\mathcal{W}$ as a witness that G does not admit a core-stable allocation. We verify $\mathcal{W}$ by simulating the ellipsoid method on G, reconstructing the transcript $\mathcal{T}$ step by step. Whenever the simulation queries an allocation $\boldsymbol{\alpha}^j$, we compute $\nu(\mathsf{N})$ and test whether $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}^j_i = \nu(\mathsf{N})$; if not, the violated constraint is the preimputation constraint. Otherwise, we read the coalition S_j from $\mathcal{W}$, compute $\nu(S_j)$ in polynomial time, and verify that

$$\nu(S_j) > \sum_{i \in S_j} \boldsymbol{\alpha}^j_i,$$

which reconstructs the violated constraint $\omega^j = \left(\sum_{i \in S_j} \boldsymbol{\alpha}_i \geq \nu(S_j)\right)$ and certifies that S_j blocks $\boldsymbol{\alpha}^j$. We supply ω^j to the simulated ellipsoid method. The verifier accepts if and only if the simulation concludes that P is empty. As each of the r steps takes polynomial time and r is polynomially bounded, the verifier runs in polynomial time. □

Finally, we consider parameterized complexity. It is straightforward that both CV and CE are FPT with respect to the number of agents $\hat{n}$, because the number of possible blocking coalitions is bounded by $2^{\hat{n}}$, their value or cost can be computed in polynomial time, and the value or cost of the grand coalition can be computed in polynomial time.

Proposition 3.50. *PCOG-MaxMatch-CV and PCOG-MinST-CV are* FPT *with respect to $\hat{n}$ and can be solved in $2^{\hat{n}} \cdot |I|^{O(1)}$ time.*

[10] Remember that the longest constraint is of the form $\sum_{i \in [\hat{n}]} \boldsymbol{\alpha}_i = \nu(\mathsf{N})$.

Proof. Recall that allocation $\boldsymbol{\alpha}$ is core-stable if and only if it satisfies $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = \nu(\mathsf{N})$ and for every $S \subseteq \mathsf{N}$, it holds that $\sum_{i \in S} \boldsymbol{\alpha}_i \geq \nu(S)$. As both a minimum spanning tree and a maximum matching can be computed in polynomial time, and there are at most $2^{\hat{n}}$ coalition constraints to check, we can verify these conditions in $2^{\hat{n}} \cdot |I|^{O(1)}$ time, i.e., in FPT-time with respect to $\hat{n}$. This concludes the proof. □

Proposition 3.51. *PCOG-MaxMatch-CE is* FPT *with respect to* $\hat{n}$.

Proof. Recall that allocation $\boldsymbol{\alpha}$ is core-stable if and only if it satisfies $\sum_{i \in \mathsf{N}} \boldsymbol{\alpha}_i = \nu(\mathsf{N})$ and for every $S \subseteq \mathsf{N}$, it holds that $\sum_{i \in S} \boldsymbol{\alpha}_i \geq \nu(S)$. If we consider every element of $\boldsymbol{\alpha}$ a variable in a linear program, this is clearly a linear program with $\hat{n}$ variables and $2^{\hat{n}}$ constraints. Recall that as a maximum size matching can be found in polynomial time [Edm65], this linear program can be constructed in FPT-time with respect to $\hat{n}$. As its size is bounded by $\hat{n}$, it can also be solved in FPT-time with respect to $\hat{n}$. This concludes the proof. □

3.6 Conclusion

In this chapter, we established a comprehensive computational complexity landscape for partitioned combinatorial optimization games (PCOGs), a natural generalization of classical combinatorial optimization games (COGs), in which each agent owns a *portion* of the input structure rather than a single element. Our analysis of four fundamental optimization problems—MinVC, MinDS, MinST, and MaxMatch—reveals intriguing complexity patterns across CV and CE problems. We note that some upper-bound techniques used in this chapter generalize beyond the problems we studied:

- The Θ_2^{P}-membership argument used for CE in Theorems 3.21 and 3.39 generalizes quite cleanly. If the decision variant of the underlying optimization problem is in NP and the value of an optimal solution is polynomially bounded, we can use logarithmically many oracle calls in order to determine the value of an optimal solution, i.e., the cost or value to be divided among the agents. If the value of an optimal solution is not polynomially bounded, we may require a larger number of oracle calls to determine the value of an optimal solution (a polynomial number of calls would place us in Δ_2^{P}). One further oracle call then suffices: By the ellipsoid method argument, the variant of the problem in which this value is part of the input lies in coNP.

- The DP-membership argument for CV used in Theorems 3.16 and 3.34 also generalizes quite directly. If the decision variant of the underlying optimization problem is in NP, we can formulate the blocking condition as a coNP condition: Does there not exist a blocking coalition? (I.e., a subset of agents together with a solution that

is beneficial for them.) The preimputation condition can then be formulated as an NP condition: Does there exist a solution whose value equals the total assigned value? Note that it need not be an optimal solution, since the blocking condition already rules out suboptimal ones: The grand coalition together with an optimal solution would form a blocking coalition.

Future work could study the algorithmic complexity of computing a core-stable allocation for PCOGs. Note that the decision version, CE, is already Θ_2^{P}-complete for Partitioned Minimum Vertex Cover Games and Partitioned Minimum Dominating Set Games and coNP-complete for Partitioned Maximum Matching Games. This implies that the first two problems are hard for the search complexity class $\mathrm{FP}^{\mathrm{NP[log]}}$—the complexity class that contains all search problems where a desirable solution (if it exists) can be found in polynomial time with logarithmically many queries to an NP-oracle [Sel94, BT96, BKT98]—but it is unclear whether this is tight since it is not clear how to find an optimal solution via logarithmically many calls to an NP-oracle. A complementary line of work is to identify additional parameters that yield fixed-parameter tractability. For the problems we studied, graph-theoretic parameters such as the maximum vertex degree or treewidth would be a good first step. Finally, our framework also invites the study of additional optimization objectives (e.g., flows, cuts, facility-location) and of stability notions beyond the core.

For the less demanding stability concept, individual rationality (IR), we found that for the four studied optimization goals, an IR allocation always exists since the corresponding games are super-additive and super-additive games always admit an individually rational allocation. However, finding an IR allocation is NP-hard whenever the underlying optimization problem is NP-hard, as finding a preimputation becomes NP-hard. Due to both a minimum spanning tree and a maximum matching being computable in polynomial time, it is straightforward that both verification and existence of IR for MinST and MaxMatch are polynomial-time solvable. Finally, our DP-completeness results (Theorems 3.16 and 3.34) for PCOG-CV extend to IR verification through a two-fold observation: (1) the hardness holds with only one agent, in which case IR is equivalent to core stability, and (2) the DP-containment proof adapts directly to IR. In the next chapter, we discuss individual rationality in the context of multi-organizational scheduling and study the existence of individually rational solutions under an additional global optimization goal.

CHAPTER 4

Multi-Organizational Scheduling

4.1 Introduction

In this chapter, we study an NTU game (see Section 2.4.1), where the organizations' utilities are determined by possible solutions of a combinatorial problem, namely scheduling. We focus on the notion of individual rationality (cf. Definition 2.46). Unlike Chapter 3[1], where a coalition's value was given through an optimal solution, in this chapter, different solutions to the underlying problems give us different utility vectors for the organizations. Real-world use cases motivate the notion of individual rationality in this setting, where multiple organizations aim to cooperate in order to generate a better schedule. This setting has been studied under the name of multi-organizational scheduling (MOS). MOS has emerged as a crucial paradigm in distributed computing environments, where organizations collaborate by sharing their computational resources to optimize job processing [PRT09]. In this model, multiple organizations, each possessing its own machines and jobs, connect their resources through a grid network to create more efficient scheduling solutions. Collaboration can potentially improve global performance metrics, such as the overall makespan ($\mathsf{C}_{\mathsf{max}}$, i.e., the completion time of the last job) and the total sum of completion times of all jobs (ΣC). A possible real-world application of such a model was discussed in Chapter 1: a set of institutes in a university, each owning a cluster of machines used by their respective members to run computer programs. These organizations may be willing to mutualize their resources in order to balance the computational load of their clusters.

However, this collaborative framework introduces strategic considerations, as each organization prioritizes its own objectives (e.g., minimizing its own makespan). From a game-theoretical perspective, if any organization were to achieve worse performance in

[1] Our setting could still be considered as a partitioned game, as each organization owns a disjoint set of machines and jobs.

the collaborative schedule compared to operating independently, it is unfair for that organization; so the organization has an incentive to withdraw from the cooperation. Such a withdrawal could cascade into disrupting other organizations' schedules. Therefore, a fundamental desideratum in our scheduling problem is individual rationality (see Definition 2.46), ensuring that no organization performs worse under cooperation than it would independently.

While individually rational schedules are guaranteed to exist (as organizations can always default to an optimal local schedule), the challenge lies in finding one that additionally optimizes global performance metrics. This chapter focuses on two fundamental metrics: the makespan and the sum of completion times. We examine scenarios where both individual organizations and the grand coalition as a whole optimize either $\mathsf{C}_{\mathsf{max}}$ or ΣC, leading to two distinct optimization problems: $\mathsf{C}_{\mathsf{max}}$-MOS and ΣC-MOS (formally defined in Section 4.2).

Main contributions. We introduce individual rationality to the multi-organizational scheduling framework. Under this fairness concept, no organization has an incentive to withdraw from collaboration since no local schedule can achieve a better performance. We systematically investigate the algorithmic complexity of two optimization problems: $\mathsf{C}_{\mathsf{max}}$-MOS and ΣC-MOS. Generally speaking, both problems are computationally hard. More precisely, the decision variant of $\mathsf{C}_{\mathsf{max}}$-MOS is Θ_2^{P}-complete[2] while that of ΣC-MOS is NP-complete.

We also present parameterized complexity analysis considering key parameters (and their combinations). They are:

- $\hat{n}$: the number of organizations,
- m: the total number of machines,
- n: the total number of jobs,
- τ: the target value of the objective function (i.e., either makespan or sum of completion times) in the decision variant,
- $\mathsf{p}_{\mathsf{max}}$: the maximum processing time of a job,
- n_{max}: the maximum number of jobs owned by an organization, and
- $\mathsf{m}_{\mathsf{max}}$: the maximum number of machines owned by an organization.

Note that we chose parameters that are studied in the literature on scheduling [MW15, MVB18], as well as parameters that are unique to the multi-organizational setting: $\hat{n}, n_{\mathsf{max}}$, and $\mathsf{m}_{\mathsf{max}}$. The latter two are localized versions of the global parameters n and m, respectively.

[2] See Definition 2.18 for the definition of Θ_2^{P}.

Among the parameterized findings, for the parameter combination $\hat{n} + \mathsf{p}_{\mathsf{max}}$, we develop a fixed-parameter tractable (FPT) algorithm for $\mathsf{C}_{\mathsf{max}}$-MOS, based on integer-linear programming (ILP; see Section 2.3.1). Our approach is based on an FPT-algorithm with respect to $\mathsf{p}_{\mathsf{max}}$ by Mnich and Wiese [MW15] for the classical problem of minimizing the makespan; this is equivalent to our model with a single organization. They proved the existence of an optimal solution that evenly distributes all jobs of the same processing time among the machines; the difference is upper-bounded by a function in $\mathsf{p}_{\mathsf{max}}$ only. This implies that there are only a small number of different types of machines. We extend this idea and show that the number of different machine types is upper-bounded by a function of $\hat{n} + \mathsf{p}_{\mathsf{max}}$. We can then introduce an integer variable for each machine type and use ILP to find an optimal solution. We modify our ILP approach to also provide an XP algorithm with respect to $\mathsf{p}_{\mathsf{max}}$.

Via a straightforward dynamic programming (DP) approach, we also demonstrate that $\mathsf{C}_{\mathsf{max}}$-MOS (resp. ΣC-MOS) is in XP with respect to m (resp. $\mathsf{m} + \mathsf{p}_{\mathsf{max}}$). For the hardness, we prove that $\mathsf{C}_{\mathsf{max}}$-MOS remains DP-hard[3] even when $\hat{n}$ is a fixed constant.

Table 4.1 summarizes our complete complexity findings.

Related work. Pascual et al. [PRT09] initiated the study of cooperation in multi-organizational scheduling, where jobs may require parallel execution across machines. They addressed the problem of minimizing the global makespan under a *local constraint* that no organization performs worse compared to a specific local schedule, computed using a heuristic. However, this constraint differs from the individual rationality we focus on in this chapter, as the heuristic-based local schedule may not be optimal, meaning organizations might still have an incentive to leave the cooperation. Pascual et al. [PRT09] showed that their problem is NP-hard and provided approximation algorithms.

Cohen et al. [CCTW11] considered the same model and proposed approximation algorithms for sequential jobs. Durand and Pascual [DP21] examined a more general setting where the local schedules are given as input and studied its approximability. Variants of these problems also allow organizations to pursue objectives beyond minimizing the makespan of their own jobs, such as minimizing the sum of job completion times [CCTW11] or the energy required to schedule jobs [CCR14]. Other studies relaxed the individual rationality constraint, allowing organizations to accept schedules where their makespan increases, provided the increase is within a given factor [OII09, OII12, CGL$^+$13, CDMT11]. Rzadca [Rza07] introduced the notion of self-reliance, and Skowron and Rzadca [SR14] employed cooperative game theory in multi-organizational scheduling, but using Shapley values as a measure of fairness. As mentioned earlier, our definition of individual rationality is stronger as it compares each organization's outcome to its optimal local schedule, in line with standard individual rationality definitions in coalition formation games.

[3]See Definition 2.17 for a definition of DP.

Table 4.1: (Parameterized) complexity of the makespan and the sum-of-completion-times variants of MOS. Results in green are tractable, results in red are intractable ("-c" = complete, "-h" = hard); within a cell, two comma-separated entries give a lower and an upper bound, and "?" marks an open case. "DP-h" means the problem remains DP-hard even if the value of the corresponding parameter is a constant. "NP-c" for the parameter combination $n_{\max} + \mathsf{m}_{\max}$ means that the decision variant is contained in NP when either of the parameters is a constant and it remains NP-hard even if both parameters have values bounded by a constant. Results obtained in prior work are cited in the same cell: The NP-completeness for $n_{\max} + \mathsf{m}_{\max}$ follows rather directly from Cohen et al. [CCTW11]. All other results are obtained in this thesis. References [T x], [P x], and [C x] point to Theorem, Proposition, and Corollary x of this thesis. Note that all omitted two-parameter combinations either have one parameter subsumed by the other, or have the result follow directly from another.

	$\mathsf{C}_{\max}$-MOS		ΣC-MOS	
Decision variant	Θ_2^{P}-c	[T4.4]	NP-c	[T4.32]
# organizations $\hat{n}$	DP-h, ?	[P4.20]	W[1]-h, ?	[P4.34]
# machines m	W[1]-h, XP	[P4.31] [JKMS13]	?	?
# jobs n	FPT	[P4.22]	FPT	[P4.36]
target value τ	FPT	[C4.28]	FPT	[C4.37]
max. processing time $\mathsf{p}_{\max}$	?, XP	[T4.30]	?	?
max. # of jobs/machines $n_{\max} + \mathsf{m}_{\max}$	NP-c	[P4.21] [CCTW11]	NP-c	[T4.32]
$\mathsf{p}_{\max} + \hat{n}$	FPT	[T4.27]	?	?
$\mathsf{p}_{\max} + \mathsf{m}$	FPT	[T4.27]	?, XP	[P4.38]
$\mathsf{p}_{\max} + n_{\max}$	FPT	[C4.29]	?	?

Parameterized complexity has recently gained attention in scheduling [MVB18]. In the classical setting (with a single organization), the problem of minimizing the makespan is shown to be FPT with respect to the maximum processing time of a task $\mathsf{p}_{\max}$ [MW15, KKM20]. Multi-organizational cooperation has also been studied in other contexts, such as matching [BKPP19, GMP12] (see Chapter 3 for our treatment in the PCOG framework) and kidney exchange [SU13, AR12, AR14, KVPS21].

Our model is also related to scheduling with processing set restrictions, surveyed by Leung and Li [LL08, LL16]. There, each job comes with a set of machines it may be assigned to, and no valid schedule may assign it elsewhere. Processing sets thus restrict where a job can be scheduled, whereas individual rationality restricts when it can be scheduled. Biró and McDermid [BM14] studied a correspondence between matching with sizes and parallel-machine scheduling with processing set restrictions. They show that if processing sets are nested—meaning that any pair of processing sets either does not intersect or one is a (not necessarily proper) subset of the other—then a greedy algorithm solves the scheduling problem in polynomial time if all processing times are

taken from $\{1, 2, 4, \ldots, 2^k\}$ for a constant k. Our individual rationality constraints have a similar structure: In the case of the makespan, every job has to be completed within the time window between 0 and the optimal local-makespan of its organization. These time windows are nested, a property that we will exploit in Observation 4.24.

4.2 Preliminaries

For background and definitions from scheduling, see Section 2.5. An instance of *MOS* is a tuple $\langle \mathcal{O}, (M_i)_{i\in[\hat{n}]}, (J_i)_{i\in[\hat{n}]}, (\mathsf{p}^i_j)_{i\in[\hat{n}],j\in[n_i]}\rangle$, where $\mathcal{O}$ denotes a set of organizations with $|\mathcal{O}| = \hat{n}$ such that for each $i \in [\hat{n}]$,

- M_i denotes a non-empty set of $\mathsf{m}_i{=}|M_i|$ identical machines,
- J_i denotes a set of $n_i = |J_i|$ non-preemptive (i.e., each job has to be completely processed before another job can start on the same machine) and sequential jobs α^i_j, i.e., $J_i = \{\alpha^i_1, \ldots, \alpha^i_{n_i}\}$ and
- for each $j \in [n_i]$, $\mathsf{p}^i_j \in \mathbb{N}$ denotes the processing time of the j-th job[4] in J_i, i.e., p^i_j is the processing time of α^i_j,

all associated with organization O_i.

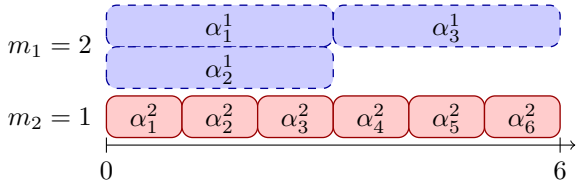
Throughout, we assume that I denotes an instance of MOS of the form $\langle \mathcal{O}, (M_i)_{i\in[\hat{n}]}, (J_i)_{i\in[\hat{n}]}, (\mathsf{p}^i_j)_{i\in[\hat{n}],j\in[n_i]}\rangle$. Moreover, we denote by $\mathcal{M}$ the set of all machines, i.e., $\mathcal{M} = \bigcup_{i\in[\hat{n}]} M_i$, by $\mathcal{J}$ the set of all jobs, i.e., $\mathcal{J} = \bigcup_{i\in[\hat{n}]} J_i$, by n the total number of jobs, and finally by m the total number of machines.

A schedule σ is a *local* schedule for *organization* $O_i \in \mathcal{O}$ if it is feasible (cf. Definition 2.54) and for every job $\alpha^i_j \in J_i$ it holds that $\mathsf{m}^i_j(\sigma) \in M_i$. We will generally just use schedules to refer to feasible schedules when it is clear from the context.

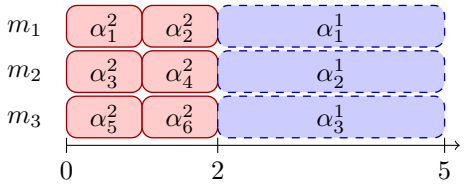
Optimal local makespan and sum of completion times. We extend the scheduling notation from Section 2.5 to the multi-organizational setting. The makespan (resp. the Σ-time) of *organization* $O_i \in \mathcal{O}$ with respect to σ is $\mathsf{C}^i_{\mathsf{max}}(\sigma) = \mathsf{C}_{\mathsf{max}}(\sigma, J_i)$ (resp. $\mathsf{C}^i_{\Sigma}(\sigma) = \mathsf{C}_{\Sigma}(\sigma, J_i)$). We omit the second argument $\mathcal{J}'$ when we refer to the makespan (resp. Σ-time) of all jobs, i.e., $\mathsf{C}_{\mathsf{max}}\ (\sigma) = \mathsf{C}_{\mathsf{max}}(\sigma, \mathcal{J})$ and $\Sigma C\ (\sigma) = \mathsf{C}_{\Sigma}(\sigma, \mathcal{J})$, respectively.

The *optimal local-makespan* (resp. *optimal local-Σ-time*) of *organization* O_i, denoted as $\mathsf{OPT\text{-}C}^i_{\mathsf{max}}$ (resp. $\mathsf{OPT\text{-}C}^i_{\Sigma}$) is the minimum over the makespans (resp. minimum over the Σ-times) of all *local* schedules of O_i. In other words, the optimal local-makespan (resp. optimal local-Σ-time) of an organization O_i is the minimum makespan (resp. minimum Σ-time) achievable by any schedule where all jobs of O_i are only scheduled to the machines of O_i. A schedule σ is an *optimal local schedule* for organization O_i if it is a local schedule for O_i and has makespan equal to $\mathsf{OPT\text{-}C}^i_{\mathsf{max}}$ (resp. Σ-time equal to $\mathsf{OPT\text{-}C}^i_{\Sigma}$).

[4] In this chapter, we assume that the instance is encoded in unary.



(a) Optimal local schedules. Each organization runs only its own jobs on its own machines; both have local makespan 6.



(b) A global optimal schedule σ. Pooling all machines and jobs, two unit-length jobs of O_2 precede one length-3 job of O_1 on each of the three machines, giving makespan 5.

Figure 4.1: Schedules for the instance in Example 4.1. Each job is represented by a rectangle whose length equals its processing time; jobs in the same row run on the same machine, and time runs from left to right, so a job drawn further left is processed earlier. The completion time of a job is the time at which it finishes, i.e., the right edge of its rectangle. O_1's jobs are blue and have a dashed boundary. O_2's jobs are red and have a solid boundary.

Example 4.1. *We consider an example with two organizations: O_1 and O_2. Organization O_1 owns $\mathsf{m}_1 = 2$ machines and $n_1 = 3$ jobs with processing times $\mathsf{p}_1^1 = \mathsf{p}_2^1 = \mathsf{p}_3^1 = 3$. Organization O_2 owns $\mathsf{m}_2 = 1$ machine and $n_2 = 6$ jobs with processing times $\mathsf{p}_1^2 = \mathsf{p}_2^2 = \cdots = \mathsf{p}_6^2 = 1$. Possible local schedules for the instance are drawn in Figure 4.1a.*

Individual rationality. As mentioned before, this chapter considers an NTU game in the setting of scheduling. It is a minimization game (cf. Definition 2.45), as each organization aims to minimize its local objective function. The organizations correspond to the players in this setting. In our setting, the cost of an organization O_i with respect to a feasible schedule σ is given by $\mathsf{C}_{\mathsf{max}}^i(\sigma)$ if the objective is to minimize the makespan and by $\mathsf{C}_{\Sigma}^i(\sigma)$ if the objective is to minimize the sum of completion times. We can then say that the cost function maps each subset of organizations S to the vectors of makespans (resp. sums of completion times) of all feasible schedules that utilize only jobs and machines that belong to organizations in S. As we only consider the notion of individual rationality, it suffices to consider the cost function μ for the grand coalition and the coalition consisting of singular players. In other words, it suffices to compare all global schedules to the optimal local schedules for each organization.

This lets us define individual rationality (cf. Definition 2.46) more precisely in our setting:

Definition 4.2. *A schedule σ is called individually rational if no organization is worse off compared to its optimal local makespan or local sum of completion times. More precisely, for the objective of minimizing the makespan ($\mathsf{C}_{\mathsf{max}}$), we require that $\mathsf{C}_{\mathsf{max}}^i(\sigma) \leq \mathsf{OPT}\text{-}\mathsf{C}_{\mathsf{max}}^i$ holds for each $O_i \in \mathcal{O}$, while for the objective of minimizing the sum of completion times (ΣC), we require that $\mathsf{C}_{\Sigma}^i(\sigma) \leq \mathsf{OPT}\text{-}\mathsf{C}_{\Sigma}^i$ holds for each $O_i \in \mathcal{O}$.*

Clearly, for both objectives, individually rational schedules exist, as one can compute an optimal schedule for each organization and combine them into a global one.

Example 4.3. *The local schedules displayed in Figure 4.1a are optimal local schedules for both organizations and both objectives. The optimal local makespans of both organizations are* $\mathsf{OPT\text{-}C}^1_{\mathsf{max}} = \mathsf{OPT\text{-}C}^2_{\mathsf{max}} = 6$. *By definition, this is an individually rational schedule. We consider a schedule* σ *in which two jobs of* O_2 *are first on each machine followed by one of the three jobs of* O_1*, depicted in Figure 4.1b. Then* σ *is individually rational and minimizes the overall makespan: The processing times of all jobs sum up to* 15*, and we have* 3 *machines. So the minimum makespan is at least* 5*. Moreover, the makespan of* O_1 *is* 5 *while the makespan of* O_2 *is* 2*.*

One can check that σ *is also optimal when we aim at minimizing* Σ*-time instead, with a total of* $24 = 3 + 6 + 15$*. However, it is not individually rational if* Σ*-time is the objective. Indeed, the optimal local-*Σ*-time of the organizations are* $\mathsf{OPT\text{-}C}^1_{\Sigma} = 3 + 3 + 6 = 12$ *and* $\mathsf{OPT\text{-}C}^2_{\Sigma} = 1 + 2 + 3 + 4 + 5 + 6 = 21$ *and the sum of completion times of* O_1 *in* σ *is 5+5+5=15.*

Central problems. We look at two optimization problems, which aim for an optimal solution among all individually rational schedules. In the following, let $\Omega \in \{\mathsf{C}_{\mathsf{max}}, \Sigma C\}$.

Ω-MOS
Input: An instance I of MOS.
Task: Find a schedule σ among all *individually rational* schedules for I such that $\Omega(\sigma)$ is minimum.

The decision variants, called Ω-MOS-Dec, have additionally a non-negative integer τ as input and ask whether there exists an *individually rational* schedule σ with $\Omega(\sigma) \leq \tau$.

Remarks. Note that in the classical setting (i.e., when the number of organizations is one), it is NP-hard to find a schedule with minimum makespan, whereas for the minimum sum of completion times case, it is polynomial-time solvable [Bru04]. Hence, $\mathsf{C}_{\mathsf{max}}$-MOS-Dec is contained in Σ^{P}_2 while ΣC-MOS-Dec is in NP. We will show that $\mathsf{C}_{\mathsf{max}}$-MOS-Dec is between NP and Σ^{P}_2; it is Θ^{P}_2-complete (Theorem 4.4), while ΣC-MOS-Dec is NP-complete (Theorem 4.32).

4.3 Minimizing the Makespan

4.3.1 General Complexity

We start this section by showing that $\mathsf{C}_{\mathsf{max}}$-MOS-Dec is Θ^{P}_2-complete.

Theorem 4.4. $\mathsf{C}_{\mathsf{max}}$*-MOS-Dec is* Θ^{P}_2*-complete.*

Proof. We start with the containment proof. To this end, we introduce an intermediate NP scheduling problem for MOS and show how to answer $\mathsf{C_{max}}$-MOS-Dec by making only logarithmically many calls to the NP-oracle of the newly introduced problem.

$\mathsf{C_{max}}$-MOS-LocalSchedules ($\mathsf{C_{max}}$-MOS-LS)
Input: An instance $\langle \mathcal{O}, (M_i)_{i\in[\hat{n}]}, (J_i)_{i\in[\hat{n}]}, (\mathsf{p}^i_j)_{i\in[\hat{n}],j\in[n_i]}\rangle$ of MOS and two integers τ_g and T'.
Question: Are there two schedules σ and σ_lo of all jobs such that:

1. For all $(i,j) \in [\hat{n}] \times [n_i]$: $\mathsf{m}^i_j(\sigma_\mathsf{lo}) \in M_i$,
2. $\sum_{i\in[\hat{n}]} \left(\max_{j\in[n_i]} \mathsf{C}^i_j(\sigma_\mathsf{lo})\right) \leq T'$,
3. $\left(\max_{\alpha^i_j\in\mathcal{J}} \mathsf{C}^i_j(\sigma)\right) \leq \tau_\mathsf{g}$, and
4. for all $(i,j) \in [\hat{n}] \times [n_i]$: $\mathsf{C}^i_j(\sigma) \leq \max\limits_{j'\in[n_i]} \{\mathsf{C}^i_{j'}(\sigma_\mathsf{lo})\}$?

Intuitively, this problem asks whether there is a local schedule with sum of makespans of the organizations equal to T' and a global schedule with makespan at most τ_g such that no organization has a larger makespan in the global schedule than in the local schedule.

Clearly, $\mathsf{C_{max}}$-MOS-LS is contained in NP as we can check in polynomial time whether two given schedules σ and σ_lo fulfill the four conditions stated in the problem definition. We now describe an algorithm answering the $\mathsf{C_{max}}$-MOS problem using only a logarithmic number of calls to an oracle solving $\mathsf{C_{max}}$-MOS-LS.

Let I be an instance of $\mathsf{C_{max}}$-MOS and τ the target makespan. First, we perform a binary search on $\mathsf{C_{max}}$-MOS-LS to find the minimum sum T' of makespans among all local schedules of I. We can do this because if σ_lo is a local schedule with minimum sum T' of makespans, then (I, T', T') is a yes-instance of $\mathsf{C_{max}}$-MOS-LS such that $(\sigma, \sigma_\mathsf{lo})$ with $\sigma = \sigma_\mathsf{lo}$ is a witness. Formally, we start with $T' = \mathsf{p_{max}} \cdot n$. For every NP-oracle call, we set $\tau_\mathsf{g} = T'$ and then do a binary search to find the minimum value T' towards which the instance is still a yes-instance of $\mathsf{C_{max}}$-MOS-LS. Note that a local schedule with minimum sum T' of makespans among all local schedules is also an optimal local schedule for each organization. This is because if one organization would have a smaller local makespan, then by exchanging the corresponding schedule one would get a smaller sum of makespans of all organizations.

Once the minimum sum is found, we make one last call of the NP-oracle, where we set T' to be the found minimum and $\tau_\mathsf{g} = \tau$; recall that τ is the target makespan. We answer yes if and only if the last call returns yes. The correctness follows by checking the definition. This completes the containment proof. We now turn to the hardness proof. We reduce from a Θ^P_2-complete problem consisting of comparing two ordered sets of 3-Partition[5] instances, where we assume that in each ordered set, all yes-instances

[5] See Section 2.7 for a definition.

appear before all no-instances. An instance of the Θ_2^{P}-complete problem is a yes-instance if and only if there are more 3-Partition yes-instances in the first set than in the second. We group instances by pairs, one from the first set, $\mathcal{I}$, and one from the second, $\mathcal{I}'$, and create a set of organizations for each pair. The local schedules of these organizations are such that if $\mathcal{I}'$ is a yes-instance, then $\mathcal{I}$ must also be a yes-instance to meet both individual rationality and the makespan requirement of τ.

Remember that the 3-Partition problem is NP-hard even if all integers in X are strictly larger than $B/4$ and strictly smaller than $B/2$ and even if the values of all integers are bounded by a polynomial of q.

Definition 4.5. *Given an instance $\mathcal{I}$ of the 3-Partition problem and an integer l, we denote by $(\mathcal{I})^{\otimes l}$ the instance of 3-Partition derived from $\mathcal{I}$ in which each integer in $\mathcal{I}$ and the target sum of a triplet are multiplied by l. In other words, if the input of $\mathcal{I}$ is the set $\{x_1, \ldots, x_{3q}\}$ and B as an input, then $(\mathcal{I})^{\otimes l}$ has the set $\{x'_1, \ldots, x'_{3q}\}$, where $x'_i = x_i \cdot l$ and $B' = B \cdot l$ as an input.*

Observation 4.6. *$(\mathcal{I})^{\otimes l}$ is a yes-instance of the 3-Partition problem if and only if $\mathcal{I}$ is a yes-instance of the 3-Partition problem.*

We will show Θ_2^{P}-hardness of $\mathsf{C_{max}}$-MOS-Dec by reduction from the 3-Partition Comparison problem, that we define now. For an instance $\mathcal{I}$ of 3-Partition, we define $\chi(\mathcal{I})$ as 1 if $\mathcal{I}$ is a yes-instance and 0 otherwise.

3-Partition Comparison
Input: Two sets $\mathcal{S}^{\mathcal{I}} = \{\mathcal{I}_1, \ldots, \mathcal{I}_a\}$ and $\mathcal{S}^{\mathcal{I}'} = \{\mathcal{I}'_1, \ldots, \mathcal{I}'_{a'}\}$ of instances of the 3-Partition problem.
Question: Are there strictly more yes-instances in $\mathcal{S}^{\mathcal{I}}$ than in $\mathcal{S}^{\mathcal{I}'}$?

From a direct application of Theorem 3.2 of [LM17], 3-Partition Comparison is Θ_2^{P}-complete, even if $a = a'$ and if $\chi(\mathcal{I}_1) \geq \chi(\mathcal{I}_2) \geq \cdots \geq \chi(\mathcal{I}_a)$ and $\chi(\mathcal{I}'_1) \geq \chi(\mathcal{I}'_2) \geq \cdots \geq \chi(\mathcal{I}'_a)$. We will assume in the reduction that both these conditions are fulfilled. In the reduced instance, we aim at an individually rational schedule with a makespan τ. To reach this makespan, it is necessary that each machine is "perfectly" used, i.e., the total processing time of jobs assigned to it is precisely τ. We will create gadgets for each pair of instances $\mathcal{I}_i$ and $\mathcal{I}'_{i-1}$. If $\mathcal{I}'_{i-1}$ is a no-instance, then regardless of $\mathcal{I}_i$, it will be possible to use these machines "perfectly". If $\mathcal{I}'_{i-1}$ is a yes-instance, then it will only be possible to use machines perfectly if $\mathcal{I}_i$ is also a yes-instance. In the following, intuitions are written in *italic*.

For an instance $\mathcal{I}$ of 3-Partition, we denote by $q(\mathcal{I})$ the number of triplets of the instance, by $B(\mathcal{I})$ the value that each triplet is required to sum to, and by $x_i(\mathcal{I})$ the i-th integer in $\mathcal{I}$.

We start the reduction by creating a yes-instance of 3-Partition $\mathcal{I}'_0$, with the same number of integers as $\mathcal{I}_1$ by choosing a target value B, e.g., 6, and putting only integers of value $B/3$. We also create a no-instance of 3-Partition $\mathcal{I}_{a+1}$, with the same number

of integers as $\mathcal{I}'_a$ by choosing a target value B, e.g., 18, and putting the same number of integers of value $B/3-1$ and $B/3+1$, and one of value $B/3$ if there is an odd number of integers in $\mathcal{I}'_a$. *These instances are here to complete gadgets containing, respectively, $\mathcal{I}_1$ and $\mathcal{I}'_a$. Since $\mathcal{I}'_0$ is a yes-instance, for the reduced instance to be a yes-instance, it is necessary that $\mathcal{I}_1$ is also a yes-instance. Note that this is required for the 3-Partition Comparison instance to be a yes-instance as well, since the question asks if there are strictly more yes-instances in $\mathcal{S}^{\mathcal{I}}$ than in $\mathcal{S}^{\mathcal{I}'}$. With the same idea in mind, $\mathcal{I}_{a+1}$ is a no-instance, therefore for the reduced instance to be a yes-instance, it is needed that $\mathcal{I}'_a$ is a no-instance, which is also required for the 3-Partition Comparison instance to be a yes-instance. Figure 4.2 shows the pairs of instances used to create the gadgets in the reduced instance.*

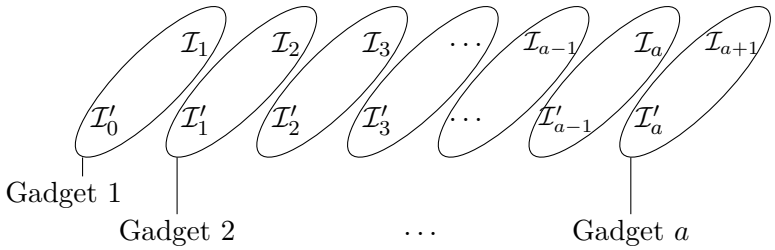


Figure 4.2: Pairing of the instances used to produce the gadgets

For all $i \in [a+1]$, we define $q_i^{\max}$ as the maximum number of triplets between $\mathcal{I}_i$ and $\mathcal{I}'_{i-1}$, i.e., $q_i^{\max} = \max\{q(\mathcal{I}_i), q(\mathcal{I}'_{i-1})\}$.

We then perform for all $i \in [a+1]$ the following modification: We replace $\mathcal{I}_i$ with $(\mathcal{I}_i)^{\otimes B(\mathcal{I}'_{i-1})}$ and $\mathcal{I}'_{i-1}$ with $(\mathcal{I}'_{i-1})^{\otimes B(\mathcal{I}_i)}$. Note that by Observation 4.6, this does not change the yes/no answer to the 3-Partition instances and therefore it does not change the yes/no answer of the 3-Partition Comparison instance. Note that with this modification, both instances have the same target sum for triplets, which is the product of the original target sums for triplets in both instances. From now on, we will denote by $\mathcal{B}_i$, the target sum of triplets for both instances $\mathcal{I}'_{i-1}$ and $\mathcal{I}_i$, i.e., $\mathcal{B}_i = B(\mathcal{I}_i) \cdot B(\mathcal{I}'_{i-1})$. If $\mathcal{B}_i$ is an odd integer, we multiply all integers in both instances as well as $\mathcal{B}_i$ by two.

We define $C(i) = k(i-1) + 2(\mathcal{B}_i + (q_i^{\max})^2\mathcal{B}_i^2)$ with $k(0) = 2$. *This value corresponds to an offset we will put in front of jobs from gadgets in order to make sure that gadgets do not interact with each other.* For all $i \in [a+1]$, we define $k(i) = C(i) + 3\mathcal{B}_i^2 q_i^{\max}/2$. *This value is an upper bound on the makespans of organizations in the i-th gadget.*

In the reduced instance the largest integer will be of value
$4k(a+1) = 4\sum_{i\in[a+1]}\left(2(\mathcal{B}_i + (q_i^{\max})^2\mathcal{B}_i^2) + 3\mathcal{B}_i^2 q_i^{\max}/2\right) + 8$. Since the value of each integer in the 3-Partition instances is bounded by a polynomial of the number of integers, this value is bounded by a polynomial of the total number of integers in the 3-Partition Comparison instance.

We are now ready to move to the construction of the reduced instance.

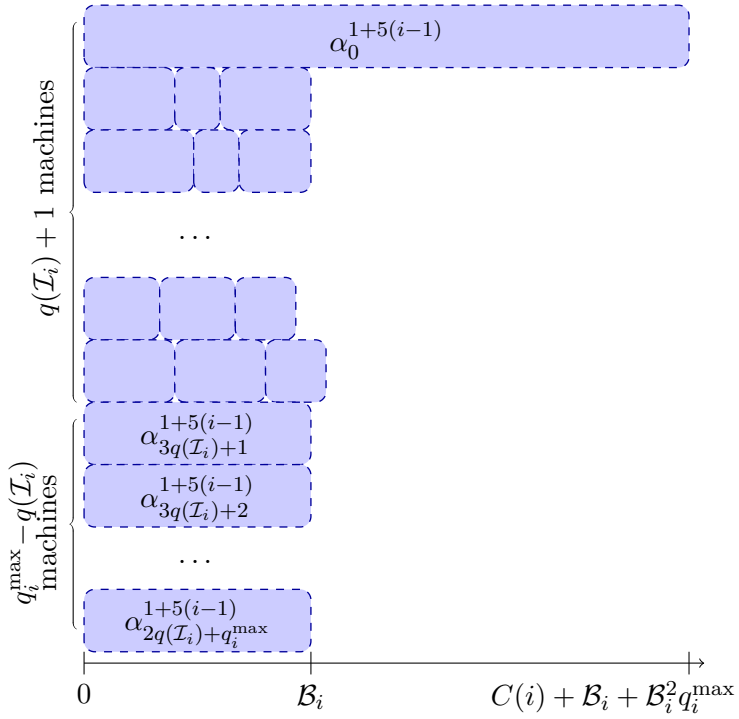


Figure 4.3: Representation of the jobs and machines of organization $O_{1+5(i-1)}$.

For all i in $[a+1]$, we create one *gadget* consisting of five organizations:

- $O_{1+5(i-1)}$: This organization owns $q_i^{\max}+1$ machines. It owns one "makespan" job $\alpha_0^{1+5(i-1)}$ of processing time $C(i)+\mathcal{B}_i+\mathcal{B}_i^2 q_i^{\max}$. It also owns a set of "integer" jobs, denoted by $\mathcal{X}_1^i$, containing:
  - 1 job for each integer in $\mathcal{I}_i$, i.e., for all $j \in [3q(\mathcal{I}_i)]$, we create a job $\alpha_j^{1+5(i-1)}$ with processing time $\mathsf{p}_j^{1+5(i-1)} = x_j(\mathcal{I}_i)$ and
  - if $q_i^{\max} > q(\mathcal{I}_i)$, it also owns $q_i^{\max} - q(\mathcal{I}_i)$ jobs $\alpha_{3q(\mathcal{I}_i)+1}^{1+5(i-1)}$ to $\alpha_{2q(\mathcal{I}_i)+q_i^{\max}}^{1+5(i-1)}$ of processing time $\mathcal{B}_i$.

  Note that regardless of whether $\mathcal{I}_i$ is a yes-instance or a no-instance, its optimal local makespan is necessarily $C(i)+\mathcal{B}_i+\mathcal{B}_i^2 q_i^{\max}$. A possible local schedule is shown in Figure 4.3.
- $O_{2+5(i-1)}$: This organization owns $q_i^{\max}$ machines. It owns a set $\mathcal{X}_2^i$ of "integer" jobs which contains:
  - One job for each integer in $\mathcal{I}'_{i-1}$. Each of these jobs has processing time equal to the value of the integer multiplied by $\mathcal{B}_i q_i^{\max}$, i.e., for all $j \in [3q(\mathcal{I}'_{i-1})]$, we create a job $\alpha_j^{2+5(i-1)}$ with processing time $\mathsf{p}_j^{2+5(i-1)} = \mathcal{B}_i q_i^{\max} \cdot x_j(\mathcal{I}'_{i-1})$.

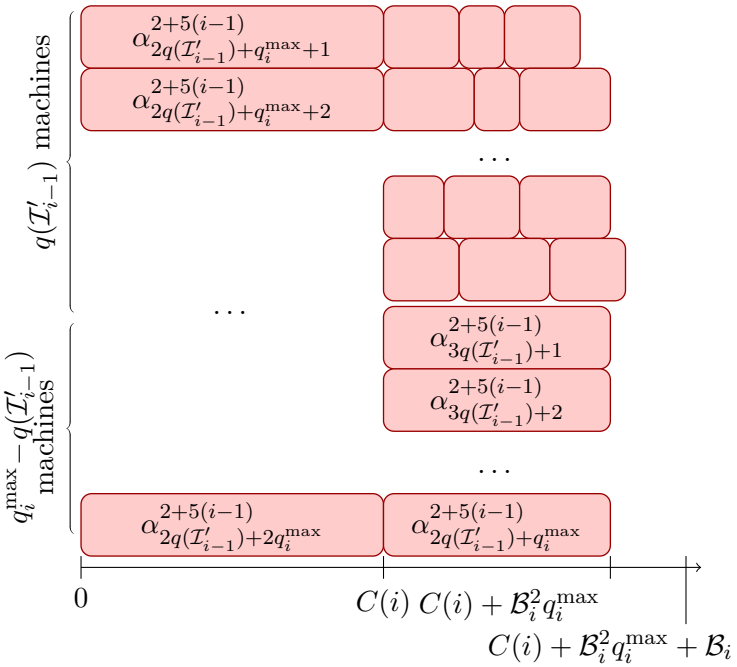


Figure 4.4: Representation of the jobs and machines of organization $O_{2+5(i-1)}$.

– If $q_i^{\max} > q(\mathcal{I}'_{i-1})$, it also owns $q_i^{\max} - q(\mathcal{I}'_{i-1})$ jobs $\alpha^{2+5(i-1)}_{3q(\mathcal{I}'_{i-1})+1}$ to $\alpha^{2+5(i-1)}_{2q(\mathcal{I}'_{i-1})+q_i^{\max}}$ of processing time $\mathcal{B}_i q_i^{\max} \cdot \mathcal{B}_i$.

Organization $O_{2+5(i-1)}$ also owns $q_i^{\max}$ "offset" jobs of processing time $C(i)$ denoted by $\alpha^{2+5(i-1)}_{2q(\mathcal{I}'_{i-1})+q_i^{\max}+1}$ to $\alpha^{2+5(i-1)}_{2q(\mathcal{I}'_{i-1})+2q_i^{\max}}$. A possible local schedule is shown in Figure 4.4.

- $O_{3+5(i-1)}$: This organization, also called *gadget equalizing* organization, owns $2q_i^{\max}$ machines. It also owns $q_i^{\max}$ jobs of processing time $k(i)$, denoted by $\alpha_1^{3+5(i-1)}$ to $\alpha_{q_i^{\max}}^{3+5(i-1)}$. It also owns a set denoted by $\mathcal{E}_3^i$ of $q_i^{\max}$ $\left(k(i) - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})\right)/\mathcal{B}_i$ "equalizing" jobs of processing time $\mathcal{B}_i$. *These jobs allow us to "fill" up to $q_i^{\max}$ machines up to $k(i)$ which will be necessary to meet the target makespan.* A possible local schedule is shown in Figure 4.5.

- $O_{4+5(i-1)}$: This organization owns $q_i^{\max}$ machines and $q_i^{\max}$ "offset" jobs of processing time $C(i)$, denoted by $\alpha_1^{4+5(i-1)}$ to $\alpha_{q_i^{\max}}^{4+5(i-1)}$. A possible local schedule is shown in Figure 4.6.

- $O_{5+5(i-1)}$: This organization owns $q_i^{\max}$ machines and $q_i^{\max}$ jobs of processing time 1, denoted by $\alpha_1^{5+5(i-1)}$ to $\alpha_{q_i^{\max}}^{5+5(i-1)}$. A possible local schedule is shown in Figure 4.7. We note here that all jobs, except the one created for organizations $O_{5+5(i-1)}$, $i \in [a+1]$, are of processing time strictly larger than 1.

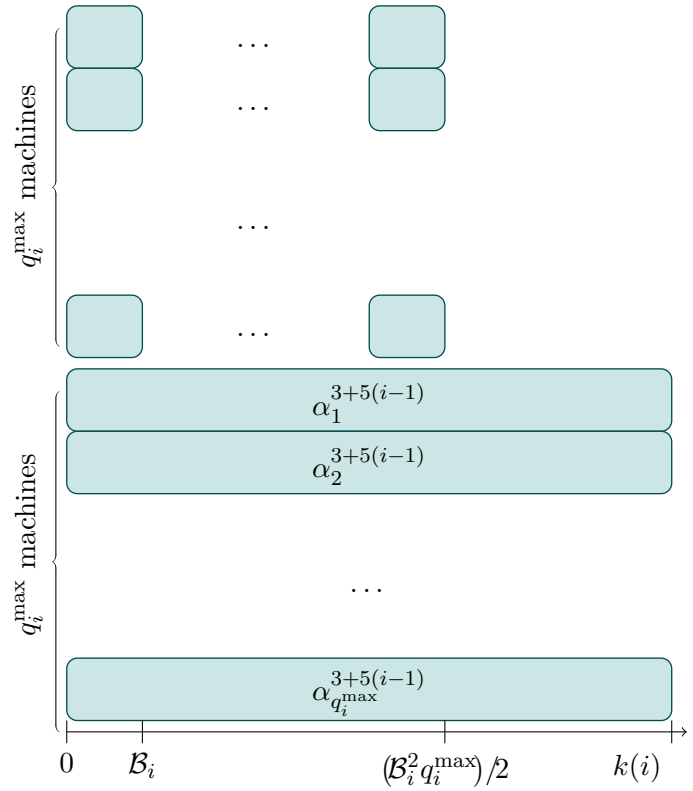


Figure 4.5: Representation of the jobs and machines of organization $O_{3+5(i-1)}$.

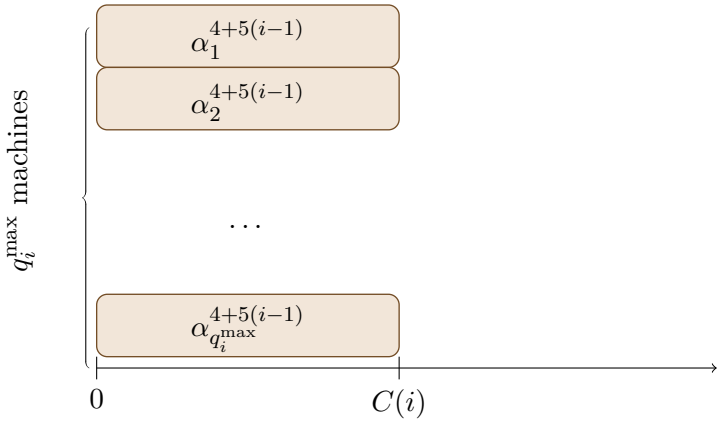


Figure 4.6: Representation of the jobs and machines of organization $O_{4+5(i-1)}$.

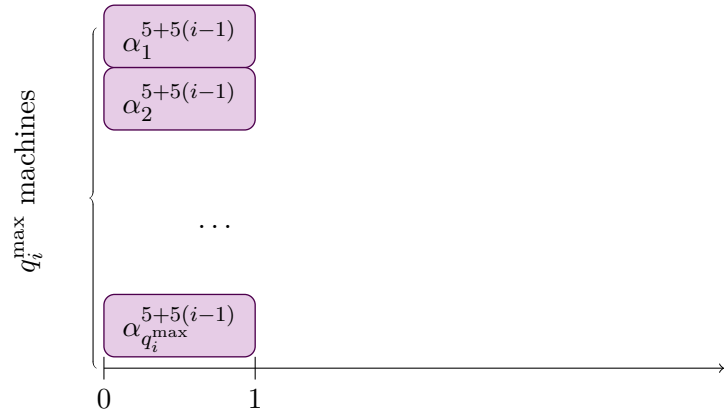


Figure 4.7: Representation of the jobs and machines of organization $O_{5+5(i-1)}$.

We complete the construction by creating one *global equalizing* organization O_{5a+6} with many long jobs of specific processing time. We set $\tau = 4(k(a+1))$. Organization O_{5a+6} owns one machine and:

- $1 + \sum_{i \in [a+1]} 3q_i^{\max}$ jobs of processing time τ,
- for all $i \in [a+1]$ one job of processing time $\tau - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})$,
- for all $i \in [a+1]$, $2q_i^{\max}$ jobs of processing time $\tau - k(i)$, and
- for all $i \in [a+1]$ $q_i^{\max}$ jobs of processing time $\tau - (C(i) + 1)$

Note that this organization has $1 + \sum_{i \in [a+1]} 3q_i^{\max} + 1 + 2q_i^{\max} + q_i^{\max} = 1 + \sum_{i \in [a+1]} 1 + 6q_i^{\max}$ jobs and that the instance has in total the same number of machines as each gadget has 5 organizations having in total $1 + 6q_i^{\max}$ machines and the global equalizing organization has one machine.

Each of the jobs of the global equalizing organization matches a machine from one of the gadgets. These very large jobs will leave a precise amount of time for machines to process the jobs of the gadget, and this will require the machines to be used perfectly. However, if $\mathcal{I}'_{i-1}$ is a yes-instance, then the only way to put the jobs corresponding to $\mathcal{I}_i$ on the same machines is to form triplets of the same size, which is only possible if $\mathcal{I}_i$ is also a yes-instance.

We start with a few straightforward observations regarding the reduced instance.

Observation 4.7. *In an individually rational schedule with makespan τ, each job of processing time τ is alone on a machine.*

Claim 4.8. *In an individually rational schedule with makespan τ, the sum of the processing times of the jobs assigned to the same machine is exactly τ.*

Proof. Let us compute the total load, i.e., the sum of the processing times of all jobs. We split the computation by gadget and match the load of a gadget with corresponding jobs of the global equalizing organization. For the i-th gadget:

- One job of processing time $(C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})$, matched by the job of processing time $\tau - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})$ owned by the global equalizing organization.
- jobs of total processing time $q_i^{\max} \cdot (\mathcal{B}_i + \mathcal{B}_i^2 \cdot q_i^{\max})$, obtained from the integers of the two instances as well as $q_i^{\max}$ jobs of processing time $C(i)$ and $q_i^{\max} \left(k(i) - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})\right) / \mathcal{B}_i$ jobs of processing time $\mathcal{B}_i$. This whole set of jobs sums up to $q_i^{\max} \cdot k(i)$ and matches $q_i^{\max}$ jobs of processing time $\tau - k(i)$ of the global equalizing organization.
- $q_i^{\max}$ jobs of processing time $k(i)$, matching the remaining $q_i^{\max}$ jobs of processing time $\tau - k(i)$ of the global equalizing organization.
- $q_i^{\max}$ jobs of processing time $C(i)$ and $q_i^{\max}$ jobs of processing time 1, matching the $q_i^{\max}$ jobs of processing time $\tau - (C(i) + 1)$ of the global equalizing organization.
- additionally, the global equalizing organization owns $3q_i^{\max}$ jobs of processing time τ.

That is a total of $(6q_i^{\max} + 1)\tau$ for the i-th gadget. We complete the proof by noting that the equalizing organization also has one machine and that there is one job of processing time τ not yet accounted for. The total load is then the number of machines times τ, which implies that in a schedule of makespan τ, each machine needs to process jobs of total processing time τ precisely. (end of the proof of Claim 4.8) $\diamond$

Claim 4.9. *In an individually rational schedule of makespan τ, each machine is assigned exactly one job from the global equalizing organization. Furthermore, this job is scheduled last on this machine.*

Proof. The first part of the statement follows directly from the fact that all jobs of the global equalizing organization have a processing time of at least 3/4 of the target makespan. Indeed, the shortest job from the global equalizing organization has a processing time of $\tau - k(a+1)$, which is precisely 3/4 of τ. The second part is shown by observing that no other organizations have an optimal local makespan strictly larger than $k(a+1)$, which is 1/4 of τ. This implies that if a job from another organization were to be scheduled after a job from the global equalizing organization, its completion time would be strictly larger than the optimal local makespan of this organization, and the schedule would then not be individually rational. (end of the proof of Claim 4.9) $\diamond$

Claim 4.10. *If there exists an individually rational schedule with makespan τ, then there exists an individually rational schedule with makespan τ in which, for all $i \in [a+1]$, jobs $\alpha_0^{1+5(i-1)}$ and the job of processing time $\tau - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})$ owned by O_{5a+6} are the only jobs on one single machine.*

Proof. Let us consider a schedule σ with $\mathsf{C}_{\mathsf{max}}(\sigma) = \tau$. By Claim 4.8, each machine processes jobs of total processing time τ. Because of individual rationality, job $\alpha_0^{1+5(i-1)}$ has to be scheduled first. By Claim 4.9, the job of processing time $\tau-(C(i)+\mathcal{B}_i+\mathcal{B}_i^2\cdot q_i^{\max})$ owned by O_{5a+6} has to be scheduled last in σ, since $\alpha_0^{1+5(i-1)}$ is scheduled first and the job of processing time $\tau - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 \cdot q_i^{\max})$ is scheduled last on their respective machines. Assuming these machines are different, we can swap the set of jobs scheduled after $\alpha_0^{1+5(i-1)}$ and the job of processing time $\tau - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 \cdot q_i^{\max})$ on $\alpha_0^{1+5(i-1)}$,s machine and obtain a new schedule with the same makespan and in which no job starts earlier nor later than in σ, this new schedule is then also individually rational.

(end of the proof of Claim 4.10) $\diamond$

Using the same argument, we can assume that jobs of processing time $k(i)$ owned by organization $O_{3+5(i-1)}$ are scheduled on a machine followed by a job of processing time $\tau - k(i)$ owned by organization O_{5a+6} for all $i \in [a + 1]$.

Claim 4.11. *If there exists an individually rational schedule with makespan τ, then there exists an individually rational schedule with makespan τ in which, for all $i \in [a + 1]$ jobs $\alpha_1^{3+5(i-1)}$ to $\alpha_{q_i^{\max}}^{3+5(i-1)}$ are each scheduled first on a machine and a job of processing time $\tau - (k(i))$ owned by O_{5a+6} is scheduled afterwards on the same machine.*

Claim 4.12. *In an individually rational schedule, for all $i \in [a + 1]$, all jobs of organizations $O_{4+5(i-1)}$ and $O_{5+5(i-1)}$ are scheduled first on a machine.*

Proof. This is straightforward since the optimal local makespan of organization $O_{4+5(i-1)}$ is $C(i)$ and all of its jobs have processing time $C(i)$, so if they are delayed by at least one unit of time, individual rationality is violated. The same argument holds for $O_{5+5(i-1)}$ by replacing $C(i)$ by 1. (end of the proof of Claim 4.12) $\diamond$

Claim 4.13. *In an individually rational schedule of makespan τ, for all $i \in [a + 1]$, for all $j \in \{i + 1, \ldots, a + 1\}$, no job owned by $O_{1+5(i-1)}$, $O_{2+5(i-1)}$, and $O_{3+5(i-1)}$ can start after a job of processing time $C(j)$.*

Proof. This follows directly from the fact that the local makespans of $O_{1+5(i-1)}$, $O_{2+5(i-1)}$ and $O_{3+5(i-1)}$ are upper bounded by $C(i + 1)$ and that if $j > i$, then $C(j) \geq C(i + 1)$ by definition. (end of the proof of Claim 4.13) $\diamond$

We are now ready to prove a claim regarding the overall structure of an existing solution if the reduced instance is a yes-instance.

Claim 4.14. *If there exists an individually rational schedule of makespan τ, then there exists an individually rational schedule of makespan τ such that, for all $i \in [a + 1]$:*

- *All jobs of processing time $C(i)$ owned by $O_{4+5(i-1)}$ are assigned to a machine with a job of O_{5a+6} of processing time $\tau - k(i)$ and*

- *all jobs of processing time $C(i)$ owned by $O_{2+5(i-1)}$ are assigned to a machine with a job of O_{5a+6} of processing time $\tau - (C(i)+1)$ and a job of processing time 1.*

Proof. We prove this by recurrence over i, starting with $i = a+1$ and going down. Let us assume that there exists an individually rational schedule σ with makespan τ. By Claim 4.10, we can assume that for all $i \in [a+1]$ jobs of processing time $\tau - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})$ owned by O_{5a+6} are assigned in σ to the same machine as the job of processing time $C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max}$ owned by $O_{1+5(i-1)}$. By Claim 4.11, we can assume for all $i \in [a+1]$ that $q_i^{\max}$ jobs of processing time $\tau - k(i)$ owned by O_{5a+6} are assigned in σ to $q_i^{\max}$ different machines, each with one of the $q_i^{\max}$ jobs of processing time $k(i)$ owned by $O_{3+5(i-1)}$.

Base case: $i = a+1$**.** By Claim 4.9, we know that in σ there is exactly one job of organization O_{5a+6} on each machine. Now observe that $C(a+1) > k(a) > C(a)+1 > \cdots > k(1) > C(1)+1$. This implies that if a job of processing time $C(a+1)$ is assigned to a machine, the only jobs of O_{5a+6} that can be assigned to the same machine in σ are jobs of processing time either $\tau-(C(a+1)+\mathcal{B}_{a+1}+\mathcal{B}_{a+1}^2 q_{a+1}^{\max})$, $\tau-k(a+1)$, or $\tau-(C(a+1)+1)$. Due to the earlier assumption, the job of processing time $\tau-(C(a+1)+\mathcal{B}_{a+1}+\mathcal{B}_{a+1}^2 q_{a+1}^{\max})$ is assigned to another machine. Furthermore, since $2C(a+1) > k(a+1) > C(a+1)+1$, it is impossible for two jobs of processing time $C(a+1)$ to be assigned to the same machine with a job of O_{5a+6}. Therefore, each job of processing time $C(a+1)$ has to be assigned to a distinct machine, and the job of O_{5a+6} assigned to these machines are of processing time either $\tau - k(a+1)$ or $\tau - (C(a+1)+1)$. Now observe that all jobs of processing time exactly 1 are owned by an organization with optimal local makespan of 1. This means that jobs of processing time 1 cannot be scheduled after any job of processing time $C(a+1)$. Furthermore, jobs of processing time $C(a+1)$ owned by O_{5a+4} have to be scheduled first in σ by Claim 4.12. This implies that jobs with processing time $\tau - (C(a+1)+1)$ are necessarily assigned to machines to which a job of processing time $C(a+1)$ owned by O_{2+5a} is assigned. Furthermore, this machine necessarily starts with a job of processing time 1, then the job of processing time $C(a+1)$, and then the job owned by the global equalizing organization.

Recursion. Assume that for all $j \in \{i+1, \ldots, a+1\}$, jobs of processing time $\tau - k(j)$ and $\tau - (C(j)+1)$ are assigned to machine with jobs of processing time $C(j)$ owned by organizations $O_{5(j-1)+2}$ and $O_{5(j-1)+4}$. By Claims 4.12 and 4.13, for all $j \in \{i+1, \ldots, a\}$ jobs of processing time $C(i)$ cannot start after jobs of processing time $C(j)$. Therefore, for all $j \in \{i+1, \ldots, a\}$ no job of processing time $C(i)$ can be assigned to the same machine as jobs of processing time $\tau - k(j)$ and $\tau - (C(j)+1)$. Now observe that $C(i) > k(i-1) > C(i-1)+1 > \cdots > k(1) > C(1)+1$. This implies that if a job of processing time $C(i)$ is assigned to a machine, the only jobs of O_{5a+6} that can be assigned to the same machine in σ are jobs of processing time either $\tau - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})$, $\tau - k(i)$, or $\tau - (C(i)+1)$. Due to the earlier assumption, the job of processing time $\tau - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})$ is assigned to another machine. The remainder of the proof is analogous to the base case. (end of the proof of Claim 4.14) $\diamond$

We finally note that, in an individually rational schedule of makespan τ, because of Claim 4.13 and Claim 4.14, it follows that the integer jobs of $O_{1+5(i-1)}$ and $O_{2+5(i-1)}$ as well as the equalizing jobs of $O_{3+5(i-1)}$ have to be assigned to the same machines as the jobs of $O_{4+5(i-1)}$.

Claim 4.15. *If there exists an individually rational schedule of makespan τ, then there exists an individually rational schedule of makespan τ such that for all $i \in [a+1]$ the jobs in $\mathcal{X}_1^i$ owned by $O_{1+5(i-1)}$, in $\mathcal{X}_2^i$ owned by $O_{2+5(i-1)}$, and $\mathcal{E}_3^i$ owned by $O_{3+5(i-1)}$ are scheduled after jobs of processing time $C(i)$ owned by $O_{4+5(i-1)}$ on the same machines and before time $k(i)$.*

Proof. Let us assume that an individually rational schedule of makespan τ exists. We consider an individually rational schedule σ of makespan τ such that:

- All jobs of processing time τ are processed alone on a machine,
- for all $i \in [a+1]$ jobs $\alpha_0^{1+5(i-1)}$ and the job of processing time $\tau-(C(i)+\mathcal{B}_i+\mathcal{B}_i^2 q_i^{\max})$ owned by O_{5a+6} are the only jobs on one single machine,
- for all $i \in [a+1]$ jobs $\alpha_1^{3+5(i-1)}$ to $\alpha_{q_i^{\max}}^{3+5(i-1)}$ are each scheduled first on a machine and a job of processing time $\tau-(k(i))$ owned by O_{5a+6} is scheduled afterwards on the same machine,
- All jobs of processing time $C(i)$ owned by $O_{4+5(i-1)}$ are assigned to a machine with a job of O_{5a+6} of processing time $\tau-k(i)$, and
- all jobs of processing time $C(i)$ owned by $O_{2+5(i-1)}$ are assigned to a machine with a job of O_{5a+6} of processing time $\tau-(C(i)+1)$ and a job of processing time 1.

By Claims 4.10 and 4.11 and Claim 4.14, we know that if the reduced instance is a yes-instance, then such a schedule exists.

By Claim 4.13, jobs owned by $O_{1+5(i-1)}$, $O_{2+5(i-1)}$, and $O_{3+5(i-1)}$ and that are not assigned according to the previous assumptions, i.e., jobs from $\mathcal{X}_1^1$,$\mathcal{X}_2^1$, and $\mathcal{E}_3^1$, cannot be scheduled on any machine except the one with starting jobs of processing time $C(1)$ owned by O_4 as any other machine either processes a job of processing time $C(i)$, with $i>1$ first or is already assigned a total processing time of τ. Additionally, the total processing time of these jobs is of $q_1^{\max}(\mathcal{B}_1+\mathcal{B}_1^2 q_1^{\max})+q_1^{\max}(k(1)-(C(1)+\mathcal{B}_1+\mathcal{B}_1^2 q_1^{\max}) = q_1^{\max}(k(1)-C(1))$. This corresponds to the amount of time between the end of the jobs of processing time $C(1)$ and the beginning of the job owned by the global equalizing organization which is of processing time $\tau-k(1)$. This means that no other job can be processed on these machines.

By repeating this observation iteratively for all i from 1 to $a+1$, we obtain the claim.

(end of the proof of Claim 4.15) $\diamond$

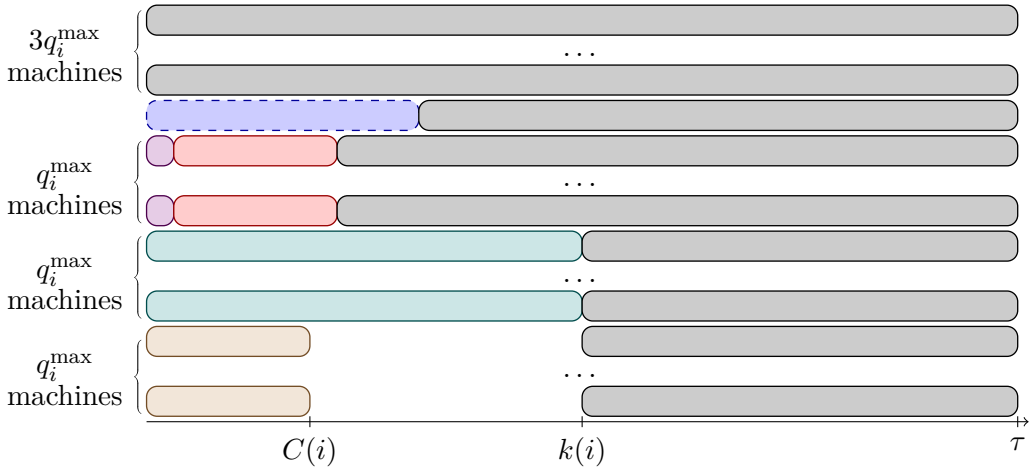


Figure 4.8: Structure of an existing individually rational schedule with makespan τ, relative to the i-th gadget, if one exists. The space left blank in the bottom machines corresponds to the time in which jobs from the sets $\mathcal{X}_1^i$,$\mathcal{X}_2^i$, and $\mathcal{E}_3^i$ have to be scheduled. Jobs in black are owned by the global equalizing organization.

Figure 4.8 shows the overall structure described by Observation 4.7 and Claims 4.10, 4.11, 4.14 and 4.15.

We will now show that the reduced instance is a yes-instance if and only if the 3-Partition Comparison instance is a yes-instance.

For the "only if" direction, to show that the reduced instance is a yes-instance only if the 3-Partition Comparison instance is a yes-instance, we prove a claim.

Claim 4.16. *If the reduced instance is a yes-instance, then for all $i \in [a+1]$, if $\mathcal{I}'_{i-1}$ is a yes-instance, then $\mathcal{I}_i$ is also a yes-instance.*

Proof. Let us assume that the reduced instance is a yes-instance, i.e., there exists an individually rational schedule of makespan τ. Let us consider for the sake of contradiction that there exists an $i \in [a+1]$ such that $\mathcal{I}'_{i-1}$ is a yes-instance and $\mathcal{I}_i$ is a no-instance.

Let us call σ an individually rational schedule in which:

- All jobs of processing time τ are processed alone on a machine,
- for all $i \in [a+1]$ jobs $\alpha_0^{1+5(i-1)}$ and the job of processing time $\tau-(C(i)+\mathcal{B}_i+\mathcal{B}_i^2 q_i^{\max})$ owned by O_{5a+6} are the only jobs on one single machine,
- for all $i \in [a+1]$ jobs $\alpha_1^{3+5(i-1)}$ to $\alpha_{q_i^{\max}}^{3+5(i-1)}$ are each scheduled first on a machine and a job of processing time $\tau - k(i)$ owned by O_{5a+6} is scheduled afterwards on the same machine,

- all jobs of processing time $C(i)$ owned by $O_{4+5(i-1)}$ are assigned to a machine with a job of O_{5a+6} of processing time $\tau - k(i)$,
- all jobs of processing time $C(i)$ owned by $O_{2+5(i-1)}$ are assigned to a machine with a job of O_{5a+6} of processing time $\tau - (C(i)+1)$ and a job of processing time 1, and
- for all $i \in [a+1]$ the jobs owned by $O_{1+5(i-1)}$, $O_{2+5(i-1)}$, and $O_{3+5(i-1)}$ except the job $\alpha_0^{1+5(i-1)}$, the jobs of processing time $C(i)$ owned by $O_{2+5(i-1)}$ and the jobs of processing time $k(i)$ owned by $O_{3+5(i-1)}$ are scheduled after jobs of processing time $C(i)$ owned by $O_{4+5(i-1)}$ on the same machines and before time $k(i)$.

By Claims 4.10, 4.11, 4.14 and 4.15 and Observation 4.7, we know that if the reduced instance is a yes-instance, then such a schedule exists.

Now, since $\mathcal{I}'_{i-1}$ is a yes-instance, it means the optimal local makespan of $O_{2+5(i-1)}$ is precisely $C(i) + \mathcal{B}_i^2 q_i^{\max}$. Indeed, a schedule with such a makespan can be obtained by putting one job of processing time $C(i)$ on each machine and dividing the jobs from $\mathcal{X}_2^i$ according to a 3-Partition of the corresponding integers in $\mathcal{I}'_{i-1}$. Since σ is individually rational, all jobs owned by $O_{2+5(i-1)}$ and scheduled on the machines starting with the jobs of processing time $C(i)$ of $O_{4+5(i-1)}$ are completed by time $C(i) + \mathcal{B}_i^2 q_i^{\max}$. As the total completion time of these jobs is of $q_i^{\max} \cdot \mathcal{B}_i^2 q_i^{\max}$, this means that no other job can be processed on these machines before time $C(i) + \mathcal{B}_i^2 q_i^{\max}$. Now, because of individual rationality, each job of $O_{1+5(i-1)}$ is completed in σ at the latest at time $C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max}$. This means that all jobs of $\mathcal{X}_1^i$ are processed between time $C(i) + \mathcal{B}_i^2 q_i^{\max}$ and $C(i) + \mathcal{B}_i^2 q_i^{\max} + \mathcal{B}_i$. It follows that each machine processes these jobs for precisely $\mathcal{B}_i$ units of time. Since jobs owned by $O_{1+5(i-1)}$ are either of processing time $\mathcal{B}_i$ (if $\mathcal{I}'_{i-1}$ contains more integers than $\mathcal{I}_i$), or of processing time matching the value of the integers of $\mathcal{I}_i$, such a schedule can only be obtained if each machine processes either one job of processing time $\mathcal{B}_i$ or three jobs with total processing time $\mathcal{B}_i$. It is easy to see that the integers corresponding to the sets of jobs processed by the same machine sum to $\mathcal{B}_i$ and therefore, the corresponding integers form valid triplets for the instance. Since all jobs can be grouped into triples of total processing time $\mathcal{B}_i$, this means that all integers of $\mathcal{I}_i$ can be partitioned into valid triplets, and $\mathcal{I}_i$ is then a yes-instance, a contradiction. (end of the proof of Claim 4.16) $\diamond$

Since $\mathcal{I}'_0$ is a yes-instance, for the reduced instance to be a yes-instance it is required that $\mathcal{I}_1$ is a yes-instance. Any additional yes-instance in $\mathcal{S}^{\mathcal{I}'}$ requires an additional yes-instance in $\mathcal{S}^{\mathcal{I}}$ for the reduced instance to be a yes-instance. Finally, since $\mathcal{I}_{a+1}$ is a no-instance, if all instances in $\mathcal{S}^{\mathcal{I}'}$ are yes-instances, then it is impossible for the reduced instance to be a yes-instance. This completes the proof of the "only if" direction.

For the "if" direction, we now assume that the 3-Partition Comparison instance is a yes-instance. This implies that there is a $j \in [a]$ such that $\forall j' \in \{0, \ldots, j-1\}, \mathcal{I}'_{j'}$ is a yes-instance, $\forall i \in [j], \mathcal{I}_i$ is a yes-instance, and $\forall j' \in \{j, \ldots, a+1\}, \mathcal{I}'_{j'}$ is a no-instance.

By Claims 4.10, 4.11, 4.14 and 4.15, we know that if an individually rational schedule with makespan τ exists, i.e., the reduced instance is a yes-instance, then one exists in which:

- All jobs of processing time τ are processed alone on a machine,
- for all $i \in [a+1]$ jobs $\alpha_0^{1+5(i-1)}$ and the job of processing time $\tau-(C(i)+\mathcal{B}_i+\mathcal{B}_i^2 q_i^{\max})$ owned by O_{5a+6} are the only jobs on one single machine,
- for all $i \in [a+1]$ jobs $\alpha_1^{3+5(i-1)}$ to $\alpha_{q_i^{\max}}^{3+5(i-1)}$ are each scheduled first on a machine and a job of processing time $\tau-(k(i))$ owned by O_{5a+6} is scheduled afterwards on the same machine,
- all jobs of processing time $C(i)$ owned by $O_{4+5(i-1)}$ are assigned to a machine with a job of O_{5a+6} of processing time $\tau-k(i)$,
- all jobs of processing time $C(i)$ owned by $O_{2+5(i-1)}$ are assigned to a machine with a job of O_{5a+6} of processing time $\tau-(C(i)+1)$ and a job of processing time 1, and
- for all $i \in [a+1]$ the jobs owned by $O_{1+5(i-1)}$, $O_{2+5(i-1)}$, and $O_{3+5(i-1)}$ except the job $\alpha_0^{1+5(i-1)}$, the jobs of processing time $C(i)$ owned by $O_{2+5(i-1)}$ and the jobs of processing time $k(i)$ owned by $O_{3+5(i-1)}$ are scheduled after jobs of processing time $C(i)$ owned by $O_{4+5(i-1)}$ on the same machines and before time $k(i)$.

We therefore start building a schedule σ following the first five properties above and provide details regarding the sixth. If no schedule fulfilling these properties exists, then the reduced instance is a no-instance. One can see that each machine has a total load of precisely τ, assuming that the jobs from $\mathcal{X}_1^i$,$\mathcal{X}_2^i$, and $\mathcal{E}_3^i$ can all be scheduled between time $C(i)$ and time $k(i)$ on machines processing jobs from $O_{4+5(i-1)}$.

For all $i \in [j]$, both $\mathcal{I}'_{i-1}$ and $\mathcal{I}_i$ are yes-instances, therefore the optimal local makespan of $O_{2+5(i-1)}$ is $C(i)+\mathcal{B}_i^2 q_i^{\max}$.

- jobs from $\mathcal{X}_2^i$ owned by $O_{2+5(i-1)}$ are scheduled in σ in the same way as in the local schedule, i.e., the jobs processed by the same machine keep their starting and completion time and are moved to the same machine processing the job of processing time $C(i)$ owned by $O_{4+5(i-1)}$. Note that this does not increase the makespan of $O_{2+5(i-1)}$ in comparison to its local makespan, therefore it does not violate individual rationality.
- Since $\mathcal{I}_i$ is a yes-instance, there exists a partition of its integers into triplets such that each triplet sums to precisely $\mathcal{B}_i$. By construction $O_{1+5(i-1)}$ owns a job of processing time $x_l(\mathcal{I}_i)$ for all $l \in 3q(\mathcal{I}_i)$. We schedule the jobs with the corresponding processing time on one of the machines, starting with a job of processing time $C(i)$ owned by $O_{4+5(i-1)}$. Each of these sets of jobs starts at time $C(i)+\mathcal{B}_i^2 q_i^{\max}$

and completes at time $C(i) + \mathcal{B}_i^2 q_i^{\max} + \mathcal{B}_i$. If $q(\mathcal{I}_i) > q(\mathcal{I}'_{i-1})$, we schedule the remaining $q_i^{\max} - q(\mathcal{I}_i)$ jobs of processing time $\mathcal{B}_i$, if any, on the remaining machines during the same time window. Note that all these jobs are completed by time $C(i)+\mathcal{B}_i^2 q_i^{\max}+\mathcal{B}_i$, which is the optimal local makespan for $O_{1+5(i-1)}$, and therefore, this does not violate individual rationality.

- Finally, we schedule the $q_i^{\max}\left(k(i) - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})\right)/\mathcal{B}_i$ jobs of processing time $\mathcal{B}_i$ of $\mathcal{E}_3^i$ owned by $O_{3+5(i-1)}$ on the $q_i^{\max}$ machines starting with a job of $O_{4+5(i-1)}$ from time $C(i) + \mathcal{B}_i^2 q_i^{\max} + \mathcal{B}_i$ to time $k(i)$. Note that $k(i) - C(i) = 3/2\mathcal{B}_i^2 q_i^{\max}$, which is a multiple of $\mathcal{B}_i$. This means that it is possible to completely fill the time on all machines with jobs of processing time $\mathcal{B}_i$. All these jobs are completed at the latest at $k(i)$, which is the optimal local makespan for $O_{3+5(i-1)}$.

For all $i \in \{j+1, \ldots, a+1\}$, $\mathcal{I}'_{i-1}$ is a no-instance.

We start by showing a few properties of an optimal local schedule.

Claim 4.17. *For all $i \in \{j+1, \ldots, a+1\}$, in an optimal local schedule for $O_{2+5(i-1)}$, there is exactly one job of processing time $C(i)$ on each machine.*

Proof. Let us assume that a machine processes two jobs of processing time $C(i)$ in an optimal local schedule, by pigeonhole principle, there is one machine which does not process any. Even if one machine not processing any job of processing time $C(i)$ processes all other jobs, its total load is $\mathcal{B}_i^2 (q_i^{\max})^2$, which is strictly smaller than $C(i)$. This means that swapping one of the jobs of processing time $C(i)$ from the machine with two with the load of the machine with none would reduce the makespan, and the schedule would then not be optimal. (end of the proof of Claim 4.17) $\diamond$

Without loss of generality, we will suppose that the jobs of processing time $C(i)$ are scheduled first on each machine in an optimal local schedule.

Claim 4.18. *For all $i \in \{j+1, \ldots, a+1\}$, in an optimal local schedule for $O_{2+5(i-1)}$, one machine has a total load of at most $C(i) + \mathcal{B}_i^2 q_i^{\max} - \mathcal{B}_i q_i^{\max}$.*

Proof. Since $\mathcal{I}'_{i-1}$ is a no-instance, it is not possible to partition its integers into triplets of sum $\mathcal{B}_i^2 q_i^{\max}$. Now observe that the processing times of all the jobs of $O_{2+5(i-1)}$, except the jobs of processing time $C(i)$, are obtained by multiplying integers by $\mathcal{B}_i q_i^{\max}$. This means that the load of each machine in an optimal local schedule for $O_{2+5(i-1)}$ is of $C(i) + \mathcal{B}_i q_i^{\max} \cdot x$ where x is an integer. Since $\mathcal{I}'_{i-1}$ is a no-instance, it is impossible to partition the jobs of $O_{2+5(i-1)}$ built from the integers of $\mathcal{I}'_{i-1}$ into $q_i^{\max}$ triplets of total processing time $\mathcal{B}_i^2 q_i^{\max}$ as otherwise it would be possible to find triplets of integers in $\mathcal{I}'_{i-1}$ summing to $\mathcal{B}_i$. This means that at least one machine is assigned a set of jobs which total processing time is strictly lower than $C(i) + \mathcal{B}_i^2 q_i^{\max}$, and since its load is of $C(i) + \mathcal{B}_i q_i^{\max} \cdot x$ where x is an integer, its load is of at most $C(i) + \mathcal{B}_i^2 q_i^{\max} - \mathcal{B}_i q_i^{\max}$.

(end of the proof of Claim 4.18) $\diamond$

Claim 4.19. *For all $i \in \{j+1, \ldots, a+1\}$, in an optimal local schedule for $O_{2+5(i-1)}$, the makespan is at most $k(i)$.*

Proof. Let us consider an optimal local schedule for $O_{2+5(i-1)}$ such that there is exactly one job of processing time $C(i)$ assigned on each machine and processed first, and such that its total load is strictly greater than $k(i) = C(i) + 3/2\mathcal{B}_i^2 q_i^{\max}$. Let us consider the last job to be processed by this machine. Since the value of each integer in $\mathcal{I}'_{i-1}$ is strictly less than $B(\mathcal{I}'_{i-1})/2$, this job is of processing time strictly less than $\mathcal{B}_i^2 q_i^{\max}/2$, which means that it starts at time at least $C(i) + \mathcal{B}_i^2 q_i^{\max} + 1$. Now, by Claim 4.18, one machine in the considered local schedule has a total load strictly lower than $C(i) + \mathcal{B}_i^2 q_i^{\max}$, which means that moving the last job to this machine would lower the makespan of the schedule, contradicting its optimality. (end of the proof of Claim 4.19) $\diamond$

We are now ready to build the second part of the schedule σ.

For all $i \in \{j+1, \ldots, a+1\}$:

- jobs from $\mathcal{X}_2^i$ owned by $O_{2+5(i-1)}$ are scheduled in σ in the same way as in the optimal local schedule, i.e., the jobs processed by the same machine keep their starting and completion time and are moved to the machine processing the job of processing time $C(i)$ owned by $O_{4+5(i-1)}$. Note that this does not increase the makespan of $O_{2+5(i-1)}$ in comparison to its optimal local makespan, therefore individual rationality is not violated.

- By Claim 4.18, there exists one machine starting with a job from $O_{4+5(i-1)}$ which processes jobs from $O_{2+5(i-1)}$ to time at most $C(i) + \mathcal{B}_i^2 q_i^{\max} - \mathcal{B}_i q_i^{\max}$, we schedule in σ all jobs in $\mathcal{X}_1^i$ from $O_{1+5(i-1)}$ on this machine. As the total processing time of these jobs is $\mathcal{B}_i q_i^{\max}$, they are all completed before $C(i) + \mathcal{B}_i^2 q_i^{\max}$, which is lower than the optimal local makespan of the organization, ensuring individual rationality for the organization.

- Finally, we schedule the $q_i^{\max} \left(k(i) - (C(i) + \mathcal{B}_i + \mathcal{B}_i^2 q_i^{\max})\right) / \mathcal{B}_i$ jobs of processing time $\mathcal{B}_i$ of $O_{3+5(i-1)}$ on the $q_i^{\max}$ machines starting with a job of $O_{4+5(i-1)}$ after the jobs from $O_{2+5(i-1)}$ and $O_{1+5(i-1)}$ such that each machine has a load of $k(i)$. Note that, as seen in Claim 4.18, each job in $\mathcal{X}_2^i$ scheduled in this time interval (between $C(i)$ and $k(i)$) has a processing time which is a multiple of $\mathcal{B}_i$, and the same holds for the block of jobs in $\mathcal{X}_1^i$. Finally, the value $k(i) - C(i)$ is also a multiple of $\mathcal{B}_i$. Therefore, we can completely fill up the remaining time between $C(i)$ and $k(i)$ with jobs of processing time $\mathcal{B}_i$. All jobs from $O_{3+5(i-1)}$ complete at the latest at time $k(i)$, which is the optimal local makespan for $O_{3+5(i-1)}$, i.e., the individual rationality is not violated.

This completes the hardness proof. $\square$

By reducing from a DP-hard problem, we can show that the problem is beyond NP and coNP, even in the case where there are only two organizations. We conjecture that the problem remains Θ_2^{P}-complete in this case.

Proposition 4.20. $\mathsf{C}_{\mathsf{max}}$*-MOS is* DP*-hard even if* $\hat{n} = 2$.

Proof. We reduce from the problem 3-Partition AND NO-3-Partition which we define now. This problem can be shown to be DP-hard by a reduction from the DP-complete problem SAT-UNSAT [Pap94], using the standard reduction from SAT to 3-Partition [GJ79].

3-Partition AND NO-3-Partition
Input: 2 instances $\mathcal{I} = \{\{x_1, \ldots, x_{3q}\}, B\}$ and $\mathcal{I}' = \{\{x'_1, \ldots, x'_{3q'}\}, B'\}$ of 3-Partition
Question: Is $\mathcal{I}$ a yes-instance of 3-Partition and $\mathcal{I}'$ a no-instance of 3-Partition?

We assume both B and B' to be even; if not, we multiply the values of the integers and the target sum by 2 in any instance in which the target sum is not even. We also assume that integers in the instances are strictly larger than the target sum divided by four and strictly smaller than the target sum divided by 2, in which case the problem remains NP-hard [GJ79]. We set $f = 3B'/2 + 1$. We create an instance with 2 organizations:

- O_1 owns q machines, for all $i \in [3q]$, it owns one job of processing time $x_i \cdot f$, it also owns one job of processing time $B'/2 + 1$. Figure 4.9 displays a possible local schedule for O_1.
- O_2 owns q' machines, for all $i \in [3q']$, it owns one "short" job of processing time x'_i, it also owns q' "long" jobs of processing time $fB - 3/2B'$. Figure 4.10 displays a possible local schedule for O_2.

We set $\tau = fB$.

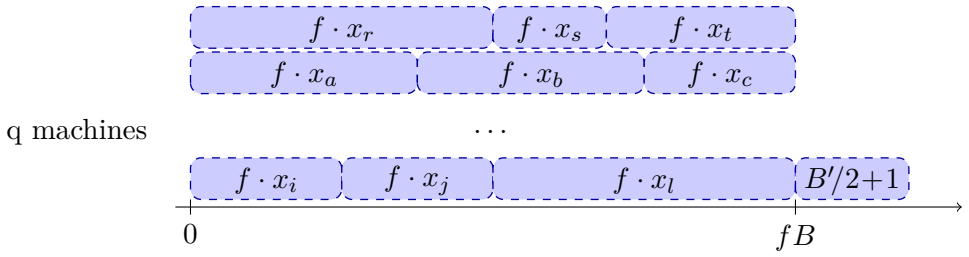


Figure 4.9: Possible local schedule for the first organization.

We start by observing that in a schedule with makespan τ, each job of processing time $fB - 3/2B'$ is on a distinct machine, indeed, if two such jobs were to be scheduled on the same machine, the total load of the machine would be at least $fB - 3B' + (3B'/2 + 1)B$,

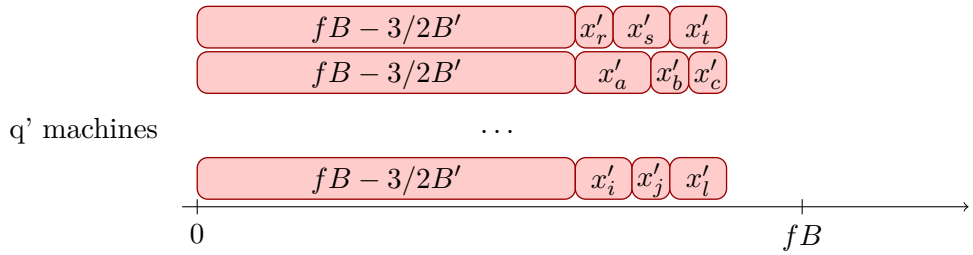


Figure 4.10: Possible local schedule for the second organization.

as B is at least 3, this would go beyond τ. Additionally, no machine can process both one job of processing time $fB - 3/2B'$ and a job from O_1, except the job of processing time $B'/2 + 1$, as it would have a load of at least $fB - 3B'/2 + (3B'/2 + 1) \cdot x$ where x is an integer, which would go beyond τ.

We now show that the 3-Partition AND NO-3-Partition instance is a yes-instance if and only if the reduced instance is a yes-instance.

For the "only if" direction, let us first assume that the 3-Partition AND NO-3-Partition instance is a yes-instance. This means that $\mathcal{I}$ is a yes-instance; therefore, there exists a partition of its integers into triplets of sum B. This implies that there is a way of splitting jobs of O_1, except the job of processing time $B'/2 + 1$, into triplets such that the total processing time of each triplet is precisely fB. We start building a schedule by scheduling each of these triplets on a distinct machine. Note that since the total load of O_1 is strictly larger than fBq and since it owns q machines, its optimal local makespan is necessarily strictly greater than fB. Since $\mathcal{I}'$ is a no-instance, it is impossible to partition its integers into triplets of sum B'. This implies that it is impossible to split short jobs of O_2 into triplets of total processing time B'. This means that its optimal local makespan is strictly larger than $fB - 3/2B' + B'$. Let us consider one optimal local schedule of O_2; each machine schedules a long job and a set of short jobs. Since it is impossible to partition the short jobs perfectly, at least one of the machines processes short jobs of total processing time at most $B' - 1$. We reproduce this local schedule on q' machines to build σ, note that this does not violate individual rationality. Finally, we put the job of processing time $B'/2 + 1$ of O_1 on a machine processing jobs of O_2 for at most $fB - 3/2B' + B' - 1$. As seen previously, this machine is guaranteed to exist. The load of this machine is at most $fB - 3/2B' + B' - 1 + B'/2 + 1 = fB = \tau$. This means that all jobs from O_1 are completed by time fB, which fulfills individual rationality.

For the "if" direction, we now assume that the reduced instance is a yes-instance. This means that there exists an individually rational schedule σ with makespan τ. As seen earlier, this schedule cannot have on the same machine either two long jobs from O_2 or one long job from O_2 and one job of O_1, except the job of processing time $B'/2 + 1$. This means that the $3q$ jobs of O_1 are scheduled on the same q machines. Since the makespan of the schedule is fB and the sum of their processing time is fBq, it means that each machine is assigned a set of jobs of total processing time exactly fB, by construction,

these sets are necessarily triplets, which implies that there exists a partition of the integers of $\mathcal{I}$ into triplets of sum B, and therefore that $\mathcal{I}$ is a yes-instance. We now look at O_2, and we suppose for the sake of contradiction that $\mathcal{I}'$ is a yes-instance. Since $\mathcal{I}'$ is a yes-instance, it is possible to partition the integers of $\mathcal{I}'$ into triplets of sum B'. Therefore, it is possible to partition the short jobs of O_2 into triplets of total processing time B'. This means that it is possible to build a local schedule in which each machine processes one long job and one triplet of jobs and has a load of $fB - B'/2$. This sets the optimal local makespan of O_2 to $fB - B'/2$. Since σ is individually rational, no job of O_2 can be completed in σ after $fB - B'/2$. It follows that the job of processing time $B'/2 + 1$ has to be scheduled either after $fB - B'/2$ or after fB on a machine with jobs from O_1. In both cases, this job would be completed after fB, a contradiction. This means that $\mathcal{I}'$ is necessarily a no-instance. This completes the proof. □

The next result shows that the problem remains NP-hard even for the case when finding an optimal local schedule for each organization is easy. The hardness persists even if each organization has only two jobs. The hardness proof follows from [CCTW11].

Proposition 4.21. *If either* $n_{\max}$ *is a constant or* $\mathsf{m}_{\max}$ *is a constant,* $\mathsf{C}_{\max}$*-MOS-Dec is contained in* NP*. It is* NP*-hard even if* $n_{\max} = 2$ *and* $\mathsf{m}_{\max} = 1$*.*

Proof. We start by observing that if $n_{\max}$ is a constant or $\mathsf{m}_{\max}$ is a constant, an optimal local schedule can be computed in polynomial time. Let us first describe an algorithm when $n_{\max}$ is a constant. If an organization has more than $n_{\max}$ machines, then an optimal local schedule can be obtained by scheduling one job on each machine. Otherwise, one can brute-force search an optimal local schedule by checking all possible schedules of the $n_{\max}$ jobs to at most $n_{\max}$ machines, which is upper-bounded by $n_{\max}^{n_{\max}}$.

Similarly, if $\mathsf{m}_{\max}$ is a constant, one can check all the up to $n_{\max}^{\mathsf{m}_{\max}}$ possible local assignments of jobs to machines for each organization. If $\mathsf{m}_{\max}$ is a constant, then this is polynomial. We can then compute an optimal local schedule in polynomial time and therefore check if a given schedule is individually rational in polynomial time. This concludes the containment proof.

For the hardness proof, as already mentioned, it follows by using the same reduction of an NP-hardness result of Cohen et al. [CCTW11, Theorem 2]. We only give a brief overview of the reduction: We again reduce from the 3-Partition problem.

Given an instance I of 3-Partition, we generate an instance I' of $\mathsf{C}_{\max}$-MOS-Dec in the following way:

- For each $i \in [3q]$, we add an organization O_i that has one machine and two jobs: α_1^i with $\mathsf{p}_1^i = (q+1)B + 11$ and α_2^i with $\mathsf{p}_2^i = (q+1)x_i + 3$.
- We add q many organizations O_{j+3q}, $j \in [q]$ that have one machine and two jobs each: α_1^{j+3q} and α_2^{j+3q} with $\mathsf{p}_1^{j+3q} = \mathsf{p}_2^{j+3q} = 1$.

We set the target makespan $\tau = (q+1)B + 11$. It can then be seen that the total processing time of the jobs is equal to $4q((q+1)B+11)$ and can only be achieved if each machine processes jobs of total processing time $(q+1)B+11$. This is possible if and only if I is a yes-instance of 3-Partition.

The reduction yields an instance where each organization has one machine and at most two jobs. Since in this case the optimal local makespan of each organization is uniquely defined and can be determined directly and a schedule satisfying the local constraint of theirs is also individually rational, the correctness follows. □

4.3.2 Algorithmic Results

We start with a fairly straightforward FPT result for the number n of jobs.

Proposition 4.22. $\mathsf{C}_{\mathsf{max}}$*-MOS is* FPT *with respect to* n *and can be solved in* $n! \cdot n^n \cdot |I|^{O(1)}$ *time.*

Proof. As we are not aware of any paper that showed the result for the standard setting, we start by showing that the local optimization problem can be solved in FPT-time with respect to n. Let $n' = |J_i|$ and $m' = |M_i|$ for organization i. We can distinguish two cases: the case where $\mathsf{m}' \geq n'$, and the case where $\mathsf{m}' < n'$. If $\mathsf{m}' \geq n'$, an optimal schedule assigns at most one job to each machine and can therefore be found in polynomial time. In the other case, one can upper-bound the number of machines by n', and therefore there are at most $(n'!) \cdot n'^{n'}$ possible schedules. Since the makespan for each of those schedules can be computed in polynomial time, it follows that the local problems are solvable in FPT-time.

We can therefore compute for each organization the time by which all their jobs have to be done. This implicitly gives each job a deadline. For the global problem, we can now use the same approach as for the local problem. We once again distinguish two cases: the case where $\mathsf{m} \geq n$, and the case where $\mathsf{m} < n$. If $\mathsf{m} \geq n$, an optimal schedule assigns each machine at most one job. This is obviously individually rational, as no organization can have a local makespan that is shorter than the longest processing time of a job that organization owns. This schedule can also be found in polynomial time by assigning one job to each machine until no jobs are left.

In the other case, there are $(n!) \cdot n^n$ possible schedules, as the number of machines is upper-bounded by n. The individual rationality can be checked in polynomial time, as for each job the deadline was computed in the preprocessing step. Therefore, since the makespan of each schedule can also be computed in polynomial time, it follows that the problem is solvable in time $n! \cdot n^n \cdot |I|^{O(1)}$ and therefore FPT with respect to n. □

Now, we turn to our main result: Theorem 4.27. As mentioned in the introduction, we extend the idea of the FPT algorithm by Mnich and Wiese [MW15]. The idea is to partition machines into groups such that all machines within a group are assigned

the same number of jobs for each processing time, as jobs of equal processing time are interchangeable. They observed that there is always a *balanced* optimal schedule. Here, *balanced* means that all jobs of the same processing time can be evenly assigned among the machines such that the difference is upper-bounded by a function in $\mathsf{p}_{\mathsf{max}}$. Due to this, the number of different groups is bounded by a function in $\mathsf{p}_{\mathsf{max}}$. Finally, one can design an ILP formulation that has an integer variable for each group specifying how many machines of that group exist in a balanced optimal schedule.

For the MOS setting, jobs belong to different organizations and may not be interchangeable, even if they have the same processing time. We circumvent this by also considering the parameter "the number $\hat{n}$ of organizations". By grouping the jobs according to the optimal local-makespan of their organization and showing that for each group and each processing time, each machine has the same number of jobs of that processing time up to a difference of a function of $\mathsf{p}_{\mathsf{max}}$, we are able to design an ILP similarly to Mnich and Wiese.

Before we show Theorem 4.27, we need some additional definitions, two auxiliary lemmas, and an observation. In the $\mathsf{C}_{\mathsf{max}}$-MOS, each organization cares only about when its last job is finished. This time cannot exceed their optimal local-makespan. We order the jobs based on the optimal local-makespan of their organization. We say two jobs α_j^i and $\alpha_{j'}^{i'}$ belong to the same *phase* if their organizations have the same optimal local-makespan, i.e., $\mathsf{OPT\text{-}C}_{\mathsf{max}}^i = \mathsf{OPT\text{-}C}_{\mathsf{max}}^{i'}$. The jobs that belong to organizations with the smallest optimal local-makespan belong to phase 1. Formally, phase 1 consists of the jobs $\{\alpha_j^i \mid \nexists i' \colon \mathsf{OPT\text{-}C}_{\mathsf{max}}^{i'} < \mathsf{OPT\text{-}C}_{\mathsf{max}}^{i}\} = \bigcup_{\arg\min \mathsf{OPT\text{-}C}_{\mathsf{max}}^i} J_i$. Similarly, the jobs that have the next smallest optimal local-makespan will be referred to as jobs in phase 2 and so on. As all the jobs belonging to a single organization belong to the same phase, it follows that the number of phases is upper-bounded by $\hat{n}$. Let $\mathsf{phase}(\alpha_j^i)$ be the phase that job α_j^i belongs to. We define the *end of phase b for machine z and schedule σ* to be $\mathsf{end}_\sigma(b, z) = \max\{0\} \cup \{\mathsf{C}_j^i(\sigma) \mid \mathsf{phase}(\alpha_j^i) \leq b \wedge \mathsf{m}_j^i(\sigma) = z\}$. Note that 0 is added to the set, as it would be possible for the set to be empty otherwise.

We start with a simple observation that jobs in an individually rational schedule can be *well-ordered*.

Definition 4.23. *A schedule σ is* well-ordered *if for each two jobs α and β that are assigned to the same machine, if α is in a phase earlier than β, then α is scheduled earlier than β.*

We can now observe that we can always find an individually rational and well-ordered schedule that minimizes the makespan.

Observation 4.24. *For each individually rational schedule σ, there exists another individually rational schedule σ' with makespan at most $\mathsf{C}_{\mathsf{max}}(\sigma)$ such that σ' is well-ordered.*

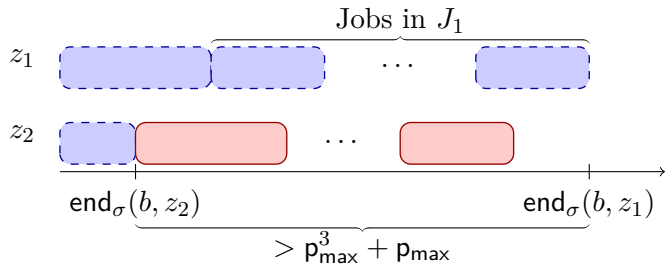


Figure 4.11: Representation of a difference in end of phase b between two machines z_1 and z_2. Jobs of phases b and before are represented in blue with dashed lines, jobs from J_2 are represented in red with solid lines.

Proof. Such a schedule σ' can be found by iteratively switching consecutive jobs if they violate the well-ordered property. Each such exchange maintains individual rationality, as a job from a later phase belongs to an organization with larger optimal local makespan, and no job except the two which were exchanged in the ordering has a different completion time after this exchange. Repeating this process exhaustively yields the desired schedule σ'. □

By Observation 4.24, from now on, we restrict our analysis to individually rational schedules that satisfy the well-ordered property. We utilize this to upper-bound the difference between completion times of each phase between two machines.

Lemma 4.25. *For each individually rational schedule σ, there exists an individually rational schedule σ' with $\mathsf{C}_{\mathsf{max}}(\sigma') \leq \mathsf{C}_{\mathsf{max}}(\sigma)$ such that for each pair of machines z_1 and z_2 and for each phase b it holds that $|\mathsf{end}_{\sigma'}(b, z_1) - \mathsf{end}_{\sigma'}(b, z_2)| \leq \mathsf{p}^3_{\mathsf{max}} + \mathsf{p}_{\mathsf{max}}$.*

Proof. Let z_1 and z_2 be two machines and σ a schedule. Without loss of generality, let $\mathsf{end}_\sigma(b, z_1) - \mathsf{end}_\sigma(b, z_2) > \mathsf{p}^3_{\mathsf{max}} + \mathsf{p}_{\mathsf{max}}$. We will argue that in this case, we can exchange jobs of phase b or lower in z_1 with jobs of a later phase in z_2, such that the difference $|\mathsf{end}_\sigma(b, z_1) - \mathsf{end}_\sigma(b, z_2)|$ shrinks. As this process can be repeated for as long as $\mathsf{end}_\sigma(b, z_1) - \mathsf{end}_\sigma(b, z_2) > \mathsf{p}^3_{\mathsf{max}} + \mathsf{p}_{\mathsf{max}}$, the lemma follows directly, by processing the phases in increasing order. Let J_1 be the set of jobs on machine z_1 that start at or after $\mathsf{end}_\sigma(b, z_2)$ and belong to phase 1 to b. We note that the sum of the processing times of jobs in J_1 is at least $\mathsf{p}^3_{\mathsf{max}}$, as the jobs are scheduled without pause on a single machine and the first job starts at $\mathsf{end}_\sigma(b, z_2) + \mathsf{p}_{\mathsf{max}}$ at the latest. Let J_2 be the set of jobs on machine z_2 that start at $\mathsf{end}_\sigma(b, z_2)$ at the earliest and are completed at $\mathsf{end}_\sigma(b, z_1)$ at the latest. For J_2 there are two options:

1. The total length of jobs in J_2 is less than $\mathsf{p}^3_{\mathsf{max}}$
2. The total length of jobs in J_2 is at least $\mathsf{p}^3_{\mathsf{max}}$.

In the first case, it follows that the difference in the makespan between the two machines is at least $\mathsf{p}_{\mathsf{max}}$. By picking an arbitrary job from J_1, making it start at $\mathsf{end}_\sigma(b, z_2)$

and shifting the other jobs on machine z_2 back, we therefore do not increase the total maximum makespan. Furthermore, this new schedule is also individually rational, as all jobs that were shifted back belong to a phase which is later than b and all jobs end at $\mathsf{end}_\sigma(b, z_1)$ at the latest. Therefore, we have shown the statement in this case.

In the second case, as the total length of jobs in J_1 and J_2 is at least $\mathsf{p}_{\mathsf{max}}^3$ each, it follows that there is some processing time p_{i_1} in J_1 such that there are at least $\mathsf{p}_{\mathsf{max}}$ many jobs of that processing time in J_1 per pigeonhole principle. Similarly, it follows that there is some processing time p_{i_2} in J_2 such that there are at least $\mathsf{p}_{\mathsf{max}}$ many jobs of that processing time in J_2. We can then exchange p_{i_1} many jobs of processing time p_{i_2} in J_2 with p_{i_2} many jobs of processing time p_{i_1} in J_1. If we place the jobs from J_2 after all jobs from phases 1 to b in J_1 we do not violate individual rationality, as they all end before $\mathsf{end}_\sigma(b, z_1)$. The jobs from J_1 will be placed directly after $\mathsf{end}_\sigma(b, z_2)$. As all jobs in J_1 start after $\mathsf{end}_\sigma(b, z_2)$, this also does not violate individual rationality. Afterward, we can once again order the jobs on each machine according to phase. This concludes the proof of this Lemma. □

The next lemma upper-bounds the number of machines of the same type and phase. Specifically, we upper- and lower-bound the number of jobs of each processing time and phase that can be assigned to a machine.

Lemma 4.26. *Given an instance I of $\mathsf{C}_{\mathsf{max}}$-MOS, let $J_{t,b}$ be the set of jobs of processing time t in phase b. Then I admits an optimal individually rational solution in which for every phase b and every distinct processing time t it holds that the number of jobs in $J_{t,b}$ scheduled on each machine is in the range $[\lfloor \frac{|J_{t,b}|}{\mathsf{m}} \rfloor - O(\mathsf{p}_{\mathsf{max}}^{\mathsf{p}_{\mathsf{max}}}), \lfloor \frac{|J_{t,b}|}{\mathsf{m}} \rfloor + O(\mathsf{p}_{\mathsf{max}}^{\mathsf{p}_{\mathsf{max}}})]$.*

Proof. This proof is a fairly minor modification of the lemma by Mnich and Wiese [MW15]. Due to Lemma 4.25 and Observation 4.24 we can assume that the machines order the jobs according to phase, and the ending times of phases differ by at most $\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p}_{\mathsf{max}}$.

Let $J_{t,b}^z$ be the set of jobs of processing time t in phase b that is scheduled on machine z. We show this lemma by showing that if the difference between the number of jobs of processing time t belonging to phase b exceeds $f(\mathsf{p}_{\mathsf{max}})$ between two machines, we can exchange jobs until the difference is at most $f(\mathsf{p}_{\mathsf{max}})$, where $f(\mathsf{p}_{\mathsf{max}}) = h(1) \cdot g(\mathsf{p}_{\mathsf{max}})$, $h(\ell) = 1 + \sum_{v=\ell+1}^{\mathsf{p}_{\mathsf{max}}} v \cdot h(v) = \frac{(\mathsf{p}_{\mathsf{max}}+1)!}{(\ell+1)!}$, and $g(\mathsf{p}_{\mathsf{max}}) = 3\mathsf{p}_{\mathsf{max}}^3 + 2\mathsf{p}_{\mathsf{max}}$. To this end, we show that there always exists an optimal solution, where $||J_{\ell,b}^z| - |J_{\ell,b}^{z'}|| \leq h(\ell) \cdot g(\mathsf{p}_{\mathsf{max}})$. We show that in case a pair of machines z, z' exists that violate this inequality for some phase b and some processing time of job ℓ we can exchange jobs of processing time ℓ with jobs of a shorter processing time in that phase, in order to satisfy this inequality. By repeating this process for all pairs of machines and all processing times, this then leads to the result we are aiming for. We show that this exchange is possible by showing that there exists a lower bound on the total length of shorter jobs.

Assume that ℓ is the largest processing time such that $|J_{\ell,b}^z| - |J_{\ell,b}^{z'}| > h(\ell) \cdot g(\mathsf{p}_{\mathsf{max}})$ for a phase b and a pair of machines z and z'. For the sake of readability, we will omit b in the

subscript, as it is fixed throughout the following inequalities:

$$\begin{aligned}
&\sum_{t=1}^{\ell-1} t \cdot |J_t^{z'}| = (\sum_{t=1}^{\mathsf{p_{max}}} t \cdot |J_t^{z'}|) - \ell \cdot |J_\ell^{z'}| - (\sum_{t=\ell+1}^{\mathsf{p_{max}}} t \cdot |J_t^{z'}|) \\
&\geq (-2(\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}) + \sum_{t=1}^{\mathsf{p_{max}}} t \cdot |J_t^{z}|) - \ell \cdot |J_\ell^{z'}| - (\sum_{t=\ell+1}^{\mathsf{p_{max}}} t \cdot |J_t^{z'}|) \\
&\geq (-2(\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}) + \sum_{t=1}^{\mathsf{p_{max}}} t \cdot |J_t^{z}|) - \ell \cdot |J_\ell^{z'}| - (\sum_{t=\ell+1}^{\mathsf{p_{max}}} t \cdot |J_t^{z}| + h(t) \cdot g(\mathsf{p_{max}})) \\
&> -2(\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}) + \sum_{t=1}^{\ell} t \cdot |J_t^{z}| + \ell(h(\ell) \cdot g(\mathsf{p_{max}}) - |J_\ell^{z}|) - \sum_{t=\ell+1}^{\mathsf{p_{max}}} t \cdot h(t) \cdot g(\mathsf{p_{max}}) \\
&= -2(\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}) + \sum_{t=1}^{\ell-1} t \cdot |J_t^{z}| + \ell(h(\ell) \cdot g(\mathsf{p_{max}})) - \sum_{t=\ell+1}^{\mathsf{p_{max}}} t \cdot h(t) \cdot g(\mathsf{p_{max}}) \\
&\geq \mathsf{p}_{\mathsf{max}}^3
\end{aligned}$$

Note that the first inequality stems from the fact that, due to Lemma 4.25, we can assume that the difference between end points of a phase is at most $\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}$, and as the end point of the previous phase can also be shifted by the same amount, we get the factor of 2.

Per pigeonhole principle, it follows that there is a processing time $p < \ell$ such that jobs of that processing time sum up to $\mathsf{p}_{\mathsf{max}}^2$. Therefore, there have to be at least $\mathsf{p_{max}}$ many of those jobs. This allows us to exchange jobs of processing time ℓ with jobs of processing time p without changing the total length of jobs in that phase on the machines, as the smallest common multiple of p and ℓ is at most $p \cdot \mathsf{p_{max}}$. Clearly, we can iterate through this process starting from the largest processing time ℓ, until the bound holds for all processing times.

We can compute $f(\mathsf{p_{max}}) = h(1) \cdot g(\mathsf{p_{max}})$. From Mnich and Wiese [MW15] it follows that this is $2^{O(\mathsf{p_{max}} \log \mathsf{p_{max}})} = O(\mathsf{p}_{\mathsf{max}}^{\mathsf{p_{max}}})$, as the only difference is that $g(\mathsf{p_{max}})$ is slightly larger than in their paper (less than a factor 3). This concludes the proof. □

The observation and lemmas allow us to search for an optimal solution with a very specific structure. This allows us to formulate an ILP with FPT running time and leads to the following theorem:

Theorem 4.27. $\mathsf{C_{max}}$*-MOS is* FPT *with respect to* $\mathsf{p_{max}} + \hat{n}$ *and therefore* $\mathsf{p_{max}} + \mathsf{m}$.

Proof. First note that $\hat{n} \leq \mathsf{m}$, as each organization has at least one machine. Therefore, it suffices to show the result for $\mathsf{p_{max}} + \hat{n}$. This approach is based on the FPT algorithm with respect to $\mathsf{p_{max}}$ that solves $(P||\mathsf{C_{max}})$ described by Mnich and Wiese [MW15]. Intuitively, the proof works by running an ILP that fixes the schedule for each phase and linking the phases together afterward.

We start by computing the number of phases by computing the optimal local makespan for each organization. By Mnich and Wiese's result, this is doable in FPT-time for each of the organizations. So computing it for all organizations is also doable in FPT-time. We now ignore the organizations and group jobs by phases as previously described. Let $[B]$ be the set of phases and t_b the latest time by which jobs of phase b must be finished. Let P_b be the set of jobs in phase $b \in [B]$ and P_b^ℓ the set of jobs with processing time ℓ in phase b. For each phase $b \in [B]$, we compute $y_b \coloneqq \sum_{\ell=1}^{\mathsf{p_{max}}} \frac{|P_b^\ell| \cdot \ell}{\mathsf{m}}$, this is the average makespan among the machines for jobs of only phase b. Note that these precomputation steps are also doable in polynomial time once the optimal local solutions for the organizations have been computed. Due to Lemma 4.25, we know that for every machine z, we can require that a machine must satisfy that $\mathsf{end}_\sigma(b,z) \in [\sum_{\ell=1}^{b} y_\ell - (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), \min\{\sum_{\ell=1}^{b} y_\ell + (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), t_b\}]$, for an optimal schedule S. Note that the left-hand side of the interval does not need a minimum, as assigning the optimal local schedule for each machine is individually rational.

Similarly we can see that the first job of the phase b must be scheduled in $[\sum_{\ell=1}^{b-1} y_\ell - (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), \min\{\sum_{\ell=1}^{b-1} y_\ell + (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), t_{b-1}\}]$, as we can assume that the jobs are ordered according to their phase due to Observation 4.24.

We can now describe the constraints and variables of the integer linear program (ILP) that solves this problem instance. Note that we do not distinguish between jobs that belong to the same phase and type in the following. We start by describing the variables:

- For each phase $b \in [B]$, each possible starting point $\mathsf{start} \in [\sum_{\ell=1}^{b-1} y_\ell - (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), \min\{\sum_{\ell=1}^{b-1} y_\ell + (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), t_{b-1}\}]$ (if $b = 1$ we fix $\mathsf{start} = 0$), each possible ending point $\mathsf{end} \in [\sum_{\ell=1}^{b} y_\ell - (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), \min\{\sum_{\ell=1}^{b} y_\ell + (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), t_b\}]$, and each vector M of length $\mathsf{p_{max}}$ that satisfies that M_t is at least $\lfloor \frac{|P_b^t|}{\mathsf{m}} \rfloor - f(\mathsf{p_{max}})$ and at most $\lfloor \frac{|P_b^t|}{\mathsf{m}} \rfloor + f(\mathsf{p_{max}})$ and $\sum_{t=1}^{\mathsf{p_{max}}} M_t \cdot t = \mathsf{end} - \mathsf{start}$, we create a variable $v_{b,\mathsf{start},\mathsf{end},M}$. Intuitively, the vector M keeps track of how many jobs of processing time t are scheduled on a machine through the entry M_t. Note that all parameters must be non-negative integers (including zero) and that M may be the zero vector. These variables must all take integer values in the range $[0, \mathsf{m}]$. Informally, the value this variable takes is the number of machines that finished the previous phase(s) at time start, have exactly the number of jobs of each processing time as in M scheduled on them in phase b, and finish phase b exactly at time end.

- We can also introduce an auxiliary variable e in order to solve the optimization problem. This is not necessary.

These variables turn out to be the only variables that are needed. In this proof, parameters will be called valid if they can form a variable as described above. We now describe the constraints that are needed.

1. We need a constraint that limits the number of machines that can be used in each phase. As the total number of machines is m, this simply means that the variables need to sum up to m for each fixed $b \in [B]$.

$$\forall\ b \in [B]\colon \sum_{\forall \text{ valid } \mathsf{start},\ \mathsf{end},\ M} v_{b,\mathsf{start},\mathsf{end},M} = \mathsf{m}$$

2. We need a constraint that makes sure that the starting times and end times of machines match between phases, such that each machine is ensured to only run one job at a time and not have any time when it is not processing any job. For the following constraint, let $C = \mathsf{p}_{\mathsf{max}}^3 + \mathsf{p}_{\mathsf{max}}$.

$$\forall\ b \in [B] \setminus \{1\}, \forall\ \mathsf{time} \in [\sum_{\ell=1}^{b-1} y_\ell - C, \min\{\sum_{\ell=1}^{b} y_\ell + C, t_b\}]\colon$$
$$\sum_{\forall \text{ valid } \mathsf{start},\ M^1} v_{b-1,\mathsf{start},\mathsf{time},M^1} = \sum_{\forall \text{ valid } \mathsf{end},\ M^2} v_{b,\mathsf{time},\mathsf{end},M^2}$$

3. We need a constraint that ensures that in each phase, all the jobs that are part of this phase are scheduled. For each processing time t we add:

$$\forall\ b \in [B]\colon \sum_{\forall \text{ valid } \mathsf{start},\mathsf{end},M} M_t \cdot v_{b,\mathsf{start},\mathsf{end},M} = |P_b^t|$$

4. In order to find an optimal solution, we need to link the auxiliary variable e to the other variables.

$$\forall\ v_{b,\mathsf{start},\mathsf{end},M}\colon \min\{v_{b,\mathsf{start},\mathsf{end},M} \cdot \mathsf{end}, \mathsf{end}\} \leq e$$

In order to solve the optimization problem for the makespan, we can then minimize e in the ILP.

We now show correctness of the ILP, by arguing that each valid schedule σ that has the form as described in Observation 4.24, Lemma 4.25, and Lemma 4.26 is a valid solution for the ILP (ignoring the minimization over e) and showing that every solution to the ILP can be transformed to a valid schedule σ.

Let σ be a valid schedule, for each phase b, start, end, and M we set $v_{b,\mathsf{start},\mathsf{end},M}$ to be equal to the number of machines that schedule jobs according to the vector M in that phase, such that $\mathsf{end}_\sigma(b-1,z) = \mathsf{start}$. This satisfies constraint 1, as each phase obviously only uses m machines. Constraint 2 is satisfied, as we set $\mathsf{end}_\sigma(b-1,z) = \mathsf{start}$, and constraint 3 is satisfied as we have a valid schedule.

For the other direction, we assign jobs phase by phase. For the first phase, we schedule $v_{1,0,\mathsf{end},M}$ many machines with exactly the job seen in M. Then in step b we choose $v_{b,\mathsf{start},\mathsf{end},M}$ many machines that satisfy that $\mathsf{end}_\sigma(b-1,z) = \mathsf{start}$ and assign the jobs in M to them. This is necessarily possible, due to constraint 2. Note that the number

of machines in this step is exactly m due to constraint 1, and all jobs in this phase are scheduled due to constraint 3.

Finally, as e only tracks the largest *end* among variables $v_{b,\mathsf{start},\mathsf{end},M} \neq 0$, it returns the makespan.

As the number of variables as well as the number of constraints is FPT with respect to $\mathsf{p}_{\mathsf{max}} + \hat{n}$, it follows that the ILP solves the problem in FPT-time with respect to $\mathsf{p}_{\mathsf{max}} + \hat{n}$ (see Theorem 2.32). This concludes the proof. □

The next two corollaries follow directly from the proof of Theorem 4.27, as the number of phases and $\mathsf{p}_{\mathsf{max}}$ can be upper-bounded by the given parameters.

Corollary 4.28. $\mathsf{C}_{\mathsf{max}}$*-MOS is* FPT *with respect to* τ*.*

Proof. It can be easily seen that τ upper-bounds $\mathsf{p}_{\mathsf{max}}$. However, the number of phases is also upper-bounded by τ, as all organizations with local makespans that exceed τ can be grouped together. This holds because for them, individual rationality is always upheld by a schedule with makespan τ. □

Corollary 4.29. $\mathsf{C}_{\mathsf{max}}$*-MOS is* FPT *with respect to* $n_{\mathsf{max}} + \mathsf{p}_{\mathsf{max}}$*.*

Proof. As we assume that each organization has at least one machine, it follows that each local makespan is upper-bounded by $n_{\mathsf{max}} \cdot \mathsf{p}_{\mathsf{max}}$. As combining all local schedules leads to an individually rational schedule, it follows that if $\tau \geq n_{\mathsf{max}} \cdot \mathsf{p}_{\mathsf{max}}$, there always exists an individually rational schedule. In the other case, τ can be upper-bounded by $(n_{\mathsf{max}} + \mathsf{p}_{\mathsf{max}})^2$, and this reduces to the previous corollary. □

Using a similar approach as for Theorem 4.27, we can show that $\mathsf{C}_{\mathsf{max}}$-MOS is XP with respect to $\mathsf{p}_{\mathsf{max}}$. The idea for the algorithm is to solve the subproblems of scheduling the first phases iteratively: We first determine all possible machine states—how many machines there are with each completion time—at the end of phase 1, then use those to determine the possible states at the end of phase 2, and so on.

Theorem 4.30. $\mathsf{C}_{\mathsf{max}}$*-MOS is* XP *with respect to* $\mathsf{p}_{\mathsf{max}}$*.*

Proof. In this proof, we group machines by type. We say that two machines share a *type* at the end of a phase if they complete their last job of that phase (or previous phases if no job in that phase was scheduled) at the same time, i.e., $\mathsf{end}_\sigma(b, z) = \mathsf{end}_\sigma(b, z')$.

Throughout, we restrict our attention to schedules that have the form described in Observation 4.24, Lemma 4.25, and Lemma 4.26, as an optimal individually rational schedule of this form always exists. We reuse the setup of the proof of Theorem 4.27. We start by computing the optimal local-makespan of each organization, which is doable in FPT-time by the result of Mnich and Wiese [MW15], and group the jobs into the phases

$[B]$ with $B \leq \hat{n}$. Let t_b be the latest time by which the jobs of phase b must be finished, P_b^ℓ be the set of jobs with processing time ℓ in phase b, and $y_b \coloneqq \sum_{\ell=1}^{\mathsf{p_{max}}} \frac{|P_b^\ell| \cdot \ell}{\mathsf{m}}$. Due to Lemma 4.25, we know that for every machine z, we can require that

$$\mathsf{end}_\sigma(b, z) \in \Big[\sum_{\ell=1}^{b} y_\ell - (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), \min\Big\{\sum_{\ell=1}^{b} y_\ell + (\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}), t_b\Big\}\Big].$$

We write T_b for the set of non-negative integers in this interval, that is, for the set of types at the end of phase b, and we set $T_0 \coloneqq \{0\}$. As the interval has length $2(\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}})$, we have $|T_b| \leq 2(\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}}) + 1$, so the number of types is bounded by a function of $\mathsf{p_{max}}$.

A *state* after phase b is a vector $\boldsymbol{s}^b = (s_\tau^b)_{\tau \in T_b}$ of non-negative integers with $\sum_{\tau \in T_b} s_\tau^b = \mathsf{m}$, where s_τ^b is the number of machines of type τ. As each entry is at most m, there are at most $(\mathsf{m}+1)^{2(\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}})+1}$ states per phase. The initial state $\boldsymbol{s}^0$ assigns all m machines to type 0. We call a state $\boldsymbol{s}^b$ *reachable* if the jobs of the phases 1 to b admit a well-ordered schedule without idle time that induces it such that, for every $b' \leq b$, every machine z satisfies $\mathsf{end}_\sigma(b', z) \in T_{b'}$ and receives, for every $t \in [\mathsf{p_{max}}]$, a number of jobs of $P_{b'}^t$ within the bounds of Lemma 4.26.

As a subroutine, we use an algorithm that decides, for a pair of states $\boldsymbol{s}^{b-1}$ and $\boldsymbol{s}^b$, whether the jobs of phase b can be assigned to the machines in such a way that $\boldsymbol{s}^{b-1}$ is turned into $\boldsymbol{s}^b$. Given this subroutine, we compute the reachable states phase by phase: Starting from $\boldsymbol{s}^0$, we test every pair consisting of a reachable state after phase $b-1$ and a possible state after phase b, and mark the possible state as reachable whenever the subroutine answers positively. After the last phase, the makespan of a schedule for a given reachable state $\boldsymbol{s}^B$ is the largest type with a non-zero entry, i.e., $\max\{\tau \in T_B \mid s_\tau^B > 0\}$. Minimizing this value over all reachable states $\boldsymbol{s}^B$ yields the optimal makespan of an individually rational schedule. One can generate an optimal schedule from this by storing a single predecessor for every reachable state, and then backtracking through them by running the ILP subroutine for the pairs.

Correctness of the algorithm follows since, due to Observation 4.24, every job of a later phase is scheduled after the end of phase b on its machine. Furthermore, every job upholds individual rationality, as we only consider end times of a phase in T_b. Since it does not matter which machine a job is scheduled on (only its completion time) due to the machines being identical, two schedules of the jobs of the phases 1 to b that induce the same state are equivalent with respect to individual rationality and the makespan. For each of the at most $\hat{n}$ many phases, we consider at most $(\mathsf{m}+1)^{2(2(\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p_{max}})+1)}$ pairs of states. Hence, this algorithm terminates in XP-time, as long as the subroutine does so as well.

It remains to describe the subroutine. Let b, $\boldsymbol{s}^{b-1}$, and $\boldsymbol{s}^b$ be fixed. We use a variant of the ILP of Theorem 4.27 that is restricted to a single phase. Note that fixing the two states does not fix a single starting point, as $\boldsymbol{s}^{b-1}$ only prescribes how many machines are of each type; the machines may therefore start phase b at any type that $\boldsymbol{s}^{b-1}$ occupies.

We start by describing the variables. For each possible starting point $\mathsf{start} \in T_{b-1}$, each possible ending point $\mathsf{end} \in T_b$, and each vector M of length $\mathsf{p_{max}}$ that satisfies that M_t is at least $\lfloor \frac{|P_b^t|}{\mathsf{m}} \rfloor - f(\mathsf{p_{max}})$ and at most $\lfloor \frac{|P_b^t|}{\mathsf{m}} \rfloor + f(\mathsf{p_{max}})$—where $f(\mathsf{p_{max}})$ is the function defined in the proof of Lemma 4.26—and $\sum_{t=1}^{\mathsf{p_{max}}} M_t \cdot t = \mathsf{end} - \mathsf{start}$, we create a variable $v_{\mathsf{start},\mathsf{end},M}$. As in Theorem 4.27, the vector M keeps track of how many jobs of processing time t are scheduled on a machine through the entry M_t. Note that all parameters must be non-negative integers (including zero). These variables must all take integer values in the range $[0, \mathsf{m}]$. Informally, the value this variable takes is the number of machines that finish phase $b-1$ at time start, have exactly the number of jobs of each processing time as in M scheduled on them in phase b, and finish phase b exactly at time end. As in the proof of Theorem 4.27, parameters will be called valid if they can form a variable as described above. We now describe the constraints that are needed.

1. We need a constraint that ensures that the machines' starting times match up with those that the state $\boldsymbol{s}^{b-1}$ provides.

$$\forall\ \mathsf{start} \in T_{b-1}\colon \sum_{\forall\ \text{valid } \mathsf{end},\ M} v_{\mathsf{start},\mathsf{end},M} = s^{b-1}_{\mathsf{start}}$$

2. We need a constraint that ensures that the machines' end times match up with those that the state $\boldsymbol{s}^{b}$ demands.

$$\forall\ \mathsf{end} \in T_{b}\colon \sum_{\forall\ \text{valid } \mathsf{start},\ M} v_{\mathsf{start},\mathsf{end},M} = s^{b}_{\mathsf{end}}$$

3. We need a constraint that ensures that all the jobs of phase b are scheduled. For each processing time t we add:

$$\sum_{\forall\ \text{valid } \mathsf{start},\mathsf{end},M} M_t \cdot v_{\mathsf{start},\mathsf{end},M} = |P_b^t|$$

Note that constraints 1 and 2 take over the role of the first two constraints in Theorem 4.27: Summing either of them over all types yields m, so the number of machines is respected, and together they force the jobs of phase b to start at the times given by $\boldsymbol{s}^{b-1}$ and to end at the times given by $\boldsymbol{s}^{b}$.

We now show correctness of the ILP, by arguing that each schedule of the jobs of phase b that turns $\boldsymbol{s}^{b-1}$ into $\boldsymbol{s}^{b}$ is a valid solution for the ILP and showing that every solution to the ILP can be transformed into such a schedule. Let σ be such a schedule. For each start, end, and M we set $v_{\mathsf{start},\mathsf{end},M}$ to be equal to the number of machines that schedule jobs according to the vector M in that phase, such that $\mathsf{end}_\sigma(b-1, z) = \mathsf{start}$ and $\mathsf{end}_\sigma(b, z) = \mathsf{end}$. Constraints 1 and 2 are satisfied, as σ induces $\boldsymbol{s}^{b-1}$ and $\boldsymbol{s}^{b}$, and constraint 3 is satisfied as σ schedules all jobs of the phase. For the other direction, we choose for each triple $(\mathsf{start}, \mathsf{end}, M)$ exactly $v_{\mathsf{start},\mathsf{end},M}$ machines of type start and assign the jobs in M to them. This is possible due to constraint 1, all jobs of the phase are

scheduled due to constraint 3, and the resulting schedule induces $\boldsymbol{s}^b$ due to constraint 2. Individual rationality is maintained, as each of these machines completes its jobs of phase b at time $\mathsf{end} \leq t_b$.

There are at most $2(\mathsf{p}_{\mathsf{max}}^3 + \mathsf{p}_{\mathsf{max}}) + 1$ choices for start and equally many for end, and at most $(2f(\mathsf{p}_{\mathsf{max}}) + 1)^{\mathsf{p}_{\mathsf{max}}}$ vectors M. As the number of variables as well as the number of constraints is therefore bounded by a function of $\mathsf{p}_{\mathsf{max}}$, it follows that the subroutine runs in FPT-time with respect to $\mathsf{p}_{\mathsf{max}}$ (see Theorem 2.32).

Therefore, the problem is solvable in XP-time with respect to $\mathsf{p}_{\mathsf{max}}$. This concludes the proof. □

Finally, we use a dynamic programming approach that keeps track of the makespans of each machine for the assigned jobs in order to show the following result:

Proposition 4.31. $\mathsf{C}_{\mathsf{max}}$*-MOS is* XP *with respect to* m *and can be solved in* $(n \cdot \mathsf{p}_{\mathsf{max}})^{\mathsf{m}+1} \cdot |I|^{O(1)}$ *time.*

Proof. We first show how to compute an optimal local schedule for an arbitrary organization in XP-time with respect to m; we call this problem *Min*$\mathsf{C}_{\mathsf{max}}$ via DP. Then, we show how to modify it to solve our problem $\mathsf{C}_{\mathsf{max}}$-MOS. For Min$\mathsf{C}_{\mathsf{max}}$, we describe the DP for a given organization O_i with jobs J_i and M_i machines. We note that the ordering of jobs on a machine does not matter, but rather their processing times. We go through the jobs of O_i in this order $\alpha_1^i, \ldots, \alpha_{|J_i|}^i$.

We maintain a dynamic table $\mathcal{D}_{\mathsf{lo}}(z_1, \ldots, z_{\mathsf{m}_i}, j) \in \{0, 1\}$, where $z_1, \ldots, z_{\mathsf{m}_i} \in [\sum_{\ell=1}^{n_i} \mathsf{p}_\ell^i]$ and $j \in [|J_i|] \cup \{0\}$. Intuitively, the table entry is 1 if it is possible to assign the first j jobs to the machines such that machine d, $d \in [\mathsf{m}_i]$ has makespan z_d. We initialize the table with $\mathcal{D}_{\mathsf{lo}}(0, \ldots, 0, 0) = 1$. We now describe the recurrence:

$$\mathcal{D}_{\mathsf{lo}}(z_1, \ldots, z_{\mathsf{m}_i}, j) = \begin{cases} 1, & \textbf{if } \exists d \in [\mathsf{m}_i] \colon \mathcal{D}_{\mathsf{lo}}(z_1, \ldots, z_d - \mathsf{p}_j^i, \ldots, z_{\mathsf{m}_i}, j-1) = 1 \\ 0, & \textbf{else} \end{cases}$$

The correctness of the recurrence is straightforward since it branches over the options of where to assign the j-th job. Each table entry can further be computed in polynomial time if all previous entries have been computed. Since the table has $(n \cdot \mathsf{p}_{\mathsf{max}})^{\mathsf{m}+1}$ entries, by finding the table entry $\mathcal{D}_{\mathsf{lo}}(z_1, \ldots, z_{m'}, |J_i|) = 1$ that minimizes $\max\{z_1, \ldots, z_{m'}\}$ an optimal schedule can be found. Therefore, we can solve Min$\mathsf{C}_{\mathsf{max}}$ and compute $\mathsf{OPT\text{-}C}_{\mathsf{max}}^i$ for each organization O_i in XP-time with respect to m. Note that we require $\mathsf{OPT\text{-}C}_{\mathsf{max}}^i$ for the individual rationality constraint.

Now, we turn to our problem. Similarly to the proof of Theorem 4.27 we divide the jobs according to phases. As a reminder, $\mathsf{phase}(\alpha_j^i)$ refers to the phase of α_j^i. We order the jobs in a way $\alpha_1, \ldots, \alpha_n$, such that the jobs satisfy $\mathsf{phase}(\alpha_1) \leq \ldots \leq \mathsf{phase}(\alpha_n)$. Note that jobs from the same phase can be ordered in an arbitrary manner. We go through the jobs in this order.

We use $t(b)$ to refer to the time by which jobs in phase b need to be done, i.e., the optimal local-makespan of the organizations whose jobs belong to this phase.

We maintain a dynamic table $\mathcal{D}(z_1, \ldots, z_{\mathsf{m}}, j) \in \{0, 1\}$, where $z_1, \ldots, z_{\mathsf{m}} \in [\sum_{i \in [\hat{n}]} \sum_{\ell=1}^{n_i} \mathsf{p}_\ell^i]$ and $j \in [\sum_{i \in [k]} n_i] \cup \{0\}$. Intuitively, the table entry is 1 if it is possible to assign the jobs $\alpha_1, \ldots, \alpha_j$ to the machines such that machine d, $d \in [\mathsf{m}]$ has a makespan of z_d and individual rationality is upheld.

We initialize the table with $\mathcal{D}(0, \ldots, 0, 0) = 1$. We now describe the recursive step:

$$\mathcal{D}(z_1, \ldots, z_{\mathsf{m}}, j) = \begin{cases} 1, & \textbf{if } \exists d \in [\mathsf{m}] \colon \mathcal{D}(z_1, \ldots, z_d - \mathsf{p}_j, \ldots, z_{\mathsf{m}}, j-1) = 1 \\ & \textbf{and } \max\{z_1, \ldots, z_{\mathsf{m}}\} \leq t(\mathsf{phase}(\alpha_j)) \\ 0 & \textbf{, else} \end{cases}$$

The correctness of the recurrence is straightforward since it branches over the options of where to assign the j-th job. We can do this, as we can assume that the optimal schedule is well-ordered due to Observation 4.24 and the ordering of jobs in the same phase on the same machine does not matter for a well-ordered schedule. Individual rationality is guaranteed, as the job must be done before the deadline due to $\max\{z_1, \ldots, z_{\mathsf{m}}\} \leq t(\mathsf{phase}(\alpha_j))$. Since the table has at most $(n \cdot \mathsf{p}_{\mathsf{max}})^{\mathsf{m}+1}$ entries and each entry can be computed in polynomial time, the whole table can be filled in $(n \cdot \mathsf{p}_{\mathsf{max}})^{\mathsf{m}+1} \cdot |I|^{O(1)}$ time. By finding the table entry $\mathcal{D}(z_1, \ldots, z_{\mathsf{m}}, n) = 1$ that minimizes $\max\{z_1, \ldots, z_{\mathsf{m}}\}$ an optimal schedule can be found. Therefore $\mathsf{C}_{\mathsf{max}}$-MOS is XP with respect to m. □

4.4 Minimizing the Sum of Completion Times

While $\mathsf{C}_{\mathsf{max}}$-MOS inherits hardness from the corresponding scheduling problem, minimizing the makespan, it is not clear that ΣC-MOS is NP-hard. Indeed, in the traditional scheduling setting, a schedule with minimum sum of completion times can be found in polynomial time [Bru04]. We show that ΣC-MOS-Dec is NP-complete, even for a constant maximum number of jobs (resp. machines) per organization.

Theorem 4.32. *ΣC-MOS-Dec is* NP*-complete. It remains* NP*-hard even if $n_{\mathsf{max}} = 3$ and $\mathsf{m}_{\mathsf{max}} = 2$.*

Proof. We start by arguing that the problem is contained in NP. We can easily check in polynomial time that a given solution is individually rational, as the optimal local sum of completion times can be computed in polynomial time, and that the total sum of completion times is indeed lower than or equal to the target.

To prove hardness, we again reduce from the 3-Partition problem.

Remember that the 3-Partition problem is NP-hard even if all the integers in X have value between $B/4$ and $B/2$; $B/4$ and $B/2$ not included. We make such an assumption for the reduction.

In the reduced instance, we create:

- q "integer" organizations labeled from O_1 to O_q. For all i in $[q]$, organization O_i owns 1 machine and three jobs $\alpha_1^i, \alpha_2^i, \alpha_3^i$ and for all j in $[3]$, $\mathsf{p}_j^i = x_{3(i-1)+j}$.
- q "triplet" organizations, labeled from O_{q+1} to O_{2q}. Each triplet organization owns 2 machines and 3 jobs of processing time B.

Optimal local schedules are obtained by running the SPT (Shortest Processing Time) list scheduling algorithm, i.e., the jobs are sorted by non-decreasing processing time and scheduled greedily [Bru04].

The local sum of completion times of each triplet organization is precisely $4B$. The local sum of completion times of the integer organizations is larger than the sum of processing times of the jobs owned by the organization, as the organization only owns 1 machine, but three jobs.

We set $\tau = 5Q$. This concludes the construction.

Intuitively, a schedule with a total sum of completion times of $5Q$ is necessarily a schedule in which all jobs owned by integer organization are scheduled first on a machine and the jobs of the triplet organization are all scheduled second on a machine. However, such a schedule is individually rational only if the sum of completion times of the jobs owned by triplet organizations is at most $4B$, as the three jobs of processing time B of any triplet organization are delayed by three jobs owned by integer organizations, the sum of the completion times of these jobs has to be at most B. As the total sum of processing times of jobs owned by integer organizations is precisely qB, each group of three jobs delaying jobs from one of the q triplet organizations needs to have processing times summing to precisely B.

We will now prove that the reduced instance is a yes-instance if and only if the 3-Partition instance is a yes-instance.

To show the "only if" direction, we first prove a claim. Note that the solution described in this claim does not necessarily satisfy individual rationality.

Claim 4.33. *A schedule minimizing the total sum of completion times assigns exactly two jobs to each machine. The job starting first on a machine is owned by an integer organization, the second by a triplet organization.*

Proof. Let us assume for the sake of contradiction that a schedule σ^* minimizing the total sum of completion times does not assign 2 jobs to each machine. Since there are $6q$ jobs and $3q$ machines, σ^* assigns at least 3 jobs to a machine and at most 1 job to another machine. We consider a machine m_1 which is assigned at least three jobs in σ^*, and a machine m_2 which is assigned at most one job in σ^*. On machine m_1, we can assume that jobs are scheduled by non-decreasing processing time, as otherwise a simple exchange between two consecutive jobs would decrease the sum of completion times. Therefore, if we call α_1, α_2 and α_3 the jobs scheduled respectively first, second and third on m_1, we can assume that $\mathsf{p}_{\alpha_1} \leq \mathsf{p}_{\alpha_2} \leq \mathsf{p}_{\alpha_3}$.

We consider the schedule σ' similar to σ^* except that α_1 is scheduled first on m_2, i.e., the job scheduled alone on m_2 in σ^* now starts after α_1; and jobs α_2 and α_3 start p_{α_1} earlier on m_1. We now argue that $\mathsf{C}_\Sigma(\sigma') < \mathsf{C}_\Sigma(\sigma^*)$. Indeed, the completion times of α_2 and α_3 are lower in σ' than in σ^* by p_{α_1} and the completion time of the job scheduled alone on m_2 in σ^* is increased by p_{α_1} in σ'. The completion time of all other jobs is the same in the two schedules. This implies that $\mathsf{C}_\Sigma(\sigma') < \mathsf{C}_\Sigma(\sigma^*)$, a contradiction.

We conclude this proof by noting that each machine is assigned two jobs, one owned by an integer organization and one from a triplet organization. Indeed, let us assume that a schedule minimizing the sum of completion times σ^* assigns two jobs owned by integer organizations to the same machine m_1. By the pigeonhole principle, it also assigns two jobs owned by triplet organizations to another machine m_2. Since all integers in X have a value strictly lower than $B/2$, the completion time of the last job on m_1 is strictly lower than B, therefore moving the second job scheduled on m_2 to the last position in m_1 would strictly decrease the sum of completion times, a contradiction. Furthermore, the job owned by the integer organization is scheduled first as it has a processing time strictly lower than the job owned by the triplet organization.

(end of the proof of Claim 4.33) $\diamond$

We now argue that the minimum sum of completion times is precisely $5Q$. Indeed, the sum of completion times of all jobs owned by integer organizations is precisely Q, as each of these jobs is scheduled first on a machine. The sum of completion times of jobs owned by triplet organizations is of $3Q$, which corresponds to the sum of their processing time, plus the delay caused by the jobs of other organizations, which sums to precisely Q; for a total of $4Q$ for jobs owned by triplet organizations and a total of $5Q$ for all jobs. This means that any schedule with a sum of completion times of $5Q$ necessarily schedules jobs of integer organizations first, one on each machine, and then jobs owned by triplet organizations afterward, one per machine.

While such a schedule obviously exists, it is not clear that an individually rational schedule can fulfill this condition. We will now show that such an individually rational schedule exists if and only if the instance of 3-Partition is a yes-instance.

For the "only if" direction, let us assume that there exists a schedule σ such that σ is individually rational and $\mathsf{C}_\Sigma(\sigma) = 5Q$. Since $\mathsf{C}_\Sigma(\sigma) = 5Q$, in σ each machine is assigned a job owned by an integer organization, scheduled first, and another job owned by a triplet organization scheduled afterwards. We suppose without loss of generality that the jobs of organization O_{q+i} are scheduled on machines $m_{3(i-1)+1}$, $m_{3(i-1)+2}$ and $m_{3(i-1)+3}$. Since σ is individually rational, the sum of completion times of jobs of O_{q+i} is at most $4B$, this is only possible if the sum of the processing times of the jobs owned by integer organizations assigned to machines $m_{3(i-1)+1}$, $m_{3(i-1)+2}$ and $m_{3(i-1)+3}$ sum to at most B. As this applies to organization O_{q+i} for all i in $[q]$ and the sum of processing times of all jobs of integer organizations sum to qB, it means that for all i in $[q]$ the sum of processing times of the three jobs owned by integer organizations assigned to $m_{3(i-1)+1}$,

$m_{3(i-1)+2}$ and $m_{3(i-1)+3}$ sum to precisely B. We call these three jobs α_j^l, $\alpha_{j'}^{l'}$, and $\alpha_{j''}^{l''}$. By construction, we have that the triplet $T_i = \{x_{3(l-1)+j} + x_{3(l'-1)+j'} + x_{3(l''-1)+j''}\}$ sums to B. This gives us a valid three-partition.

For the "if" direction, let us assume that the instance of 3-Partition is a yes-instance. We build the schedule σ as follows: For all i in $[q]$, we consider the triplet $T_i = \{x_{3(l-1)+j}, x_{3(l'-1)+j'}, x_{3(l''-1)+j''}\}$, with j, j' and j'' in [3] and put the jobs α_j^l,$\alpha_{j'}^{l'}$, and $\alpha_{j''}^{l''}$ first on machines $m_{3(i-1)+1}$,$m_{3(i-1)+2}$, and $m_{3(i-1)+3}$ respectively, we schedule the jobs of organization O_{q+i} on $m_{3(i-1)+1}$,$m_{3(i-1)+2}$, and $m_{3(i-1)+3}$ respectively after the jobs already scheduled. We now argue that the schedule σ is individually rational. It is straightforward to see that no integer organization has a larger sum of completion times in σ than in its local schedules, as all of its jobs are scheduled first on a machine. The sum of completion times of each triplet organization is precisely $4B$ by the same argument used earlier: The sum of processing times of the jobs scheduled before the jobs of any triplet organization is precisely B. This means that the three jobs of processing time B owned by any given triplet organization are delayed by B in σ, which gives a sum of completion times of $4B$ for the organization, which is precisely its local sum of completion times. Furthermore, by the previous observation about the sum of completion times, since σ assigns to each machine one job owned by an integer organization, scheduled first on the machine, and one job owned by a triplet organization scheduled afterward, the sum of completion times of σ is $5Q$. Therefore, the reduced instance is a yes-instance. □

Using a similar idea as for Theorem 4.32, we show that ΣC-MOS is W[1]-hard with respect to $\hat{n}$, i.e., it is unlikely to be in FPT according to current complexity assumptions.

Proposition 4.34. *ΣC-MOS is W[1]-hard with respect to $\hat{n}$.*

Proof. We reduce from the Unary-BinPacking problem; see Section 2.7. Remember that it is W[1]-hard with respect to the number of bins, i.e., m.

We assume $y > m$ as otherwise, we can simply put one integer in each set. Furthermore, we assume that no integer has value strictly greater than B, as otherwise the answer is necessarily no. Additionally, if an integer has value precisely B, we have to put it alone in a set and we can reduce the instance to the same with one set and the integer removed.

In the reduced instance, we create:

- 1 "integer" organization O_1, it owns m machines. For all i in $[y]$, organization O_1 owns one job α_i^1 of processing time $\mathsf{p}_i^1 = x_i$.
- m "bin" organizations, labeled from O_2 to O_{m+1}. Each bin organization owns y machines and $y + 1$ jobs of processing time B.

Optimal local schedules are obtained by running the SPT (Shortest Processing Time) list scheduling algorithm [Bru04].

The local sum of completion times of each bin organization is precisely $(y+2)B$. The local sum of completion times of the integer organization is larger than the sum of processing times of the jobs owned by the organization as the organization only owns m machines but y jobs.

We set $\tau = mB(y+1) + 2\sum_{i\in[y]} \mathsf{p}_i^1$. This concludes the construction. Note that in the reduced instance, we have $\hat{n} = f(m) = m+1$.

Intuitively, a schedule with a total sum of completion times of τ is necessarily a schedule in which all jobs owned by the integer organization are scheduled first on a machine and the jobs of the bin organization are scheduled afterward. However, such a schedule is individually rational only if the sum of completion times of the jobs owned by bin organizations is at most $(y+2)B$, which is only possible if the processing times of jobs scheduled before its jobs are no more than B, which would correspond to a feasible set of the Unary-BinPacking problem.

We will now prove that the reduced instance is a yes-instance if and only if the Unary-BinPacking instance is a yes-instance.

For the "only if" direction, we first prove a claim.

Claim 4.35. *A schedule with total sum of completion times $mB(y+1) + 2\sum_{i\in[y]} \mathsf{p}_i^1$ schedules the jobs of the integer organization first on separate machines and jobs from bin organizations later.*

Proof. First note that the sum of processing times in the instance is $mB(y+1)+\sum_{i\in[y]} \mathsf{p}_i^1$, this means that if all tasks could start at time 0, the sum of completion times would be $mB(y+1)+\sum_{i\in[y]} \mathsf{p}_i^1$. Now, the instance has $m\cdot(y+1)+y$ jobs and $m\cdot(y+1)$ machines. This means that at least y jobs are delayed by at least another job. By scheduling two jobs on the same machine, the second one is delayed by the processing time of the first one, therefore, by scheduling the jobs of the integer organization first and on disjoint machines, the delay on the jobs owned by bin organizations is in total of $\sum_{i\in[y]} \mathsf{p}_i^1$. Note that it is not possible to have a total sum of completion times lower, as this schedule could be obtained by an SPT algorithm, which is optimal for the minimization of the sum of completion times. (end of the proof of Claim 4.35) $\diamond$

While such a schedule obviously exists, it is not clear that an individually rational schedule can fulfill this condition. We will now show that if such an individually rational schedule exists, then the instance of Unary-BinPacking is a yes-instance.

Let us assume that there exists a schedule σ such that σ is individually rational and $\mathsf{C}_\Sigma(\sigma) = \tau$. Since $\mathsf{C}_\Sigma(\sigma) = \tau$, jobs of the integer organization are scheduled first, and jobs owned by bin organizations are scheduled afterward. Since σ is individually rational, the sum of completion times of jobs of any bin organization is at most $(y+2)B$. This is only possible if the sum of the processing times of the jobs owned by integer organizations assigned to machines processing one of the jobs of a bin organization sums to at most

B, as the sum of processing times of the tasks of each bin organization is $(y+1)B$. As this applies to all bin organizations, it means that for all i in $[m]$ the sum of processing times of the jobs owned by the integer organization assigned to machines processing tasks of O_{1+i} sums to at most B. We consider the set S_i of these jobs, by construction $\sum_{\alpha_j^1 \in S_i} x_j \leq B$. As all jobs owned by the integer organization are scheduled, and as there are m bin organizations, there are m sets of jobs. Therefore, we obtain m sets of integers summing to at most B. This gives us a valid bin packing.

For the "if" direction, let us assume that the instance of Unary-BinPacking is a yes-instance. We build the schedule σ as follows: For all i in $[m]$, we consider the set S_i and put the jobs $\{\alpha_j^1 | x_j \in S_i\}$ first on machines owned by O_{i+1}. We schedule the jobs of organization O_{i+1} on the same machines afterward, except for one job, which is scheduled first on a machine owned by the integer organization. We now argue that the schedule σ is individually rational. It is straightforward to see that the integer organization does not have a larger sum of completion times in σ than in its local schedule, as all of its jobs are scheduled first on a machine. The sum of completion times of each bin organization is at most $B \cdot (y+2)$ by the same argument used earlier: The sum of processing times of the jobs scheduled before the jobs of any bin organization is at most B. This means that the jobs of processing time B owned by any given bin organization are delayed by B at most in σ, which gives a sum of completion times of $B \cdot (y+2)$ for the organization, which is precisely its local sum of completion times. Furthermore, by the previous observation about the sum of completion times, since σ schedules jobs owned by the integer organization first on disjoint machines and schedules jobs of bin organizations afterward, the sum of completion times of σ is $mB(y+1) + 2\sum_{i \in [y]} \mathsf{p}_i^1$. Therefore, the reduced instance is a yes-instance. This completes the parameterized reduction. □

Similarly to Proposition 4.22, the sum of completion times case allows for an FPT result with respect to n using a simple brute-forcing approach.

Proposition 4.36. *ΣC-MOS is* FPT *with respect to n and can be solved in $n! \cdot n^n \cdot |I|^{O(1)}$ time.*

Proof. This proof works in the same way as the proof for Proposition 4.22. We can distinguish two cases: $\mathsf{m} \geq n$ and $\mathsf{m} < n$. If $\mathsf{m} \geq n$, an optimal schedule assigns at most one job to each machine again and can be found in linear time. In the other case, as $\mathsf{m} < n$ it follows that there are at most $n! \cdot n^n$ different possible schedules and since the sum of completion times can be computed in polynomial time for each of them and individual rationality can also be computed in polynomial time, the statement follows. □

As τ upper-bounds the number of jobs, the following corollary is immediate.

Corollary 4.37. *ΣC-MOS is* FPT *with respect to τ and can be solved in $\tau! \cdot \tau^\tau \cdot |I|^{O(1)}$ time.*

Proof. It holds that $\tau \geq n$, as each job has processing time at least 1 and each job has a completion time that is at least its processing time. Therefore, this result follows directly from Proposition 4.36. □

Finally, we use a dynamic programming approach similar to the one used in the proof of Proposition 4.31. As it is not possible to order the jobs according to phase, as was done for the $\mathsf{C}_{\mathsf{max}}$-MOS case, we require an additional parameter for the dynamic programming approach to function. This additional parameter allows us to keep track of the organizations' sum of completion times in a dynamic program.

Proposition 4.38. *ΣC-MOS is* XP *with respect to $\mathsf{p}_{\mathsf{max}} + \mathsf{m}$ and can be solved in $(n \cdot \mathsf{p}_{\mathsf{max}})^{4\mathsf{m}+\mathsf{m}\cdot\mathsf{p}_{\mathsf{max}}} \cdot |I|^{O(1)}$ time.*

Proof. This result can be shown by modifying the proof of Proposition 4.31 slightly. Intuitively, the modification is necessary, as it is not possible to group the jobs in phases, as an organization does not mind if a job is finished late, as long as another job finishes earlier to cancel this out.

We note that $\mathsf{m} > \hat{n}$, as each organization owns at least one machine. For each organization O_i, let J_i^ℓ denote the set of jobs of processing time ℓ. Intuitively, we will modify the DP from Proposition 4.31 by keeping track of how many jobs from each set J_i^ℓ have been assigned and the sum of completion times for each machine and organization. We compute the function $\mathcal{D}(z_1, \ldots, z_\mathsf{m}, t_1, \ldots, t_{\hat{n}}, j_1^1, \ldots, j_1^{\mathsf{p}_{\mathsf{max}}}, \ldots, j_{\hat{n}}^{\mathsf{p}_{\mathsf{max}}}) \in \{0,1\}$, where $z_1, \ldots, z_\mathsf{m}, t_1, \ldots, t_{\hat{n}} \in [(\sum_{i=1}^{\hat{n}} \sum_{\ell=1}^{n_i} \mathsf{p}_\ell^i)^2]$ and $j_i^\ell \in [|J_i^\ell|] \cup \{0\}$. Intuitively, this function will take the value 1 if it is possible to assign j_i^ℓ jobs from J_i^ℓ for all i, ℓ to the machines such that machine d has a makespan of z_d and organization i has a local sum of completion time of t_i restricted to the currently assigned jobs. We initialize the table in the following way:

$$\mathcal{D}(0, \ldots, 0) = 1$$

Intuitively, this can be read as if we assign no jobs, then all machines and organizations have a sum of completion times of 0. We now describe the recursive step:

$$\mathcal{D}(z_1, \ldots, z_\mathsf{m}, t_1, \ldots, t_{\hat{n}}, j_1^1, \ldots, j_1^{\mathsf{p}_{\mathsf{max}}}, \ldots, j_{\hat{n}}^{\mathsf{p}_{\mathsf{max}}}) =$$
$$\begin{cases} 1 & \text{, if } \exists i \in [\hat{n}], d \in [\mathsf{m}], \ell \in [\mathsf{p}_{\mathsf{max}}]\text{:} \\ & \mathcal{D}(\ldots, z_d - \ell, \ldots, t_i - z_d, \ldots, j_i^\ell - 1, \ldots) = 1 \\ 0 & \text{, else} \end{cases}$$

An optimal solution can then be found by finding a table entry with value 1, such that $j_i^\ell = |J_i^\ell|$ for all i, ℓ that satisfies that t_i is smaller than or equal to the local sum of completion times and minimizes $\sum_{i \in [\hat{n}]} t_i$.

We first show that each table entry satisfies that it is a partial solution, i.e., if only j_i^ℓ jobs in J_i^ℓ are assigned for each i, ℓ, then there is a solution such that each machine d has

makespan z_d and each organization O_i has local sum of completion times t_i. We show this inductively. The base case, where every parameter is 0, obviously holds. Then the correctness follows via the recursion, as it models a task of processing time ℓ belonging to organization O_i being assigned to machine d.

On the other hand, when given a schedule, one can find a corresponding table entry, by step by step assigning the jobs to each machine.

As the table has at most $(n \cdot \mathsf{p_{max}})^{2(\mathsf{m}+\hat{n})} \cdot n^{\mathsf{m} \cdot \mathsf{p_{max}}} \leq (n \cdot \mathsf{p_{max}})^{4\mathsf{m}+\mathsf{m} \cdot \mathsf{p_{max}}}$ entries and each entry can be computed in polynomial time, it follows that the problem is solvable in XP-time with respect to $\mathsf{p_{max}} + \mathsf{m}$. This concludes the proof. □

4.5 Conclusion

We introduced the concept of individual rationality to multi-organizational scheduling and explored the parameterized complexity of two optimization problems, $\mathsf{C_{max}}$-MOS and ΣC-MOS. For the former problem, we showed that it is in XP with respect to the maximum processing time $\mathsf{p_{max}}$. As we do not have a corresponding hardness result, an important open question remains: Is $\mathsf{C_{max}}$-MOS fixed-parameter tractable with respect to $\mathsf{p_{max}}$? Notably, the classical single-organization variant of this problem is already known to be FPT with respect to $\mathsf{p_{max}}$.

Our research opens up several promising avenues for future work. First, an immediate extension is to consider the case of parallel jobs. Second, our framework can be applied to other scheduling problems, such as those with precedence constraints or hard deadlines. Third, it would also be interesting to study scenarios where each organization provides a precomputed local schedule as input. For ΣC-MOS, the complexity remains unchanged, as optimal local schedules can be computed in polynomial time. For $\mathsf{C_{max}}$-MOS, this setting reduces the problem to within NP.

Finally, an intriguing direction is to study scenarios where the local and global objectives differ. For instance, the global objective might be to minimize ΣC, while individual rationality mandates that the makespan of each organization matches its locally optimal makespan. A related model was previously examined by Cohen et al. [CCTW11], but without considering individual rationality as a constraint.

From the game-theoretic perspective, it would be interesting to investigate the complexity of finding a solution in the core (as studied for PCOGs in Chapter 3). Similarly, studying the TU version of the game (see Section 2.4.2), where the total objective value is distributed among the organizations, could lead to interesting results.

Another interesting approach could be to study whether the gains from cooperation can be shared equitably, similarly to how the Shapley value does for TU games. Consider, for example, an individually rational schedule in which only a single organization improves compared to its optimal local schedule. All other organizations have no incentive to cooperate, as they gain nothing from doing so. An approach to fairly dividing the gains has been studied by Skowron and Rzadca [SR14] in an online setting.

In the next chapter, we again turn to core stability as our stability notion. While we studied NTU-games in this chapter and TU-games in the previous chapter, the next chapter will study both NTU- and TU-games arising in the realm of multi-winner approval voting.

CHAPTER 5

Multi-Winner Games

5.1 Introduction

In this chapter, we combine multi-winner voting (see Section 2.6) with cooperative games, namely TU- and NTU-games; see Section 2.4. Due to the novel nature of this setting, we study only the unpartitioned variant, i.e., every agent corresponds to a single voter, rather than a subset of voters as would be the case in a PCOG (see Chapter 3). The aim of this chapter is to study multi-winner voting from a cooperative games perspective. For this, we use four prominent score-based voting rules: Approval Voting (AV), Satisfaction Approval Voting (SAV), Chamberlin–Courant (CC), and Proportional Approval Voting (PAV). For their formal definitions, see Section 2.6.

Much of the multi-winner literature evaluates rules via *representativeness* and *proportionality* axioms. For example, CC and PAV satisfy *justified representation* (JR) [ABC+17]. Informally, JR says that any group large enough to "deserve" one seat and agreeing on an alternative must contain at least one voter who approves a selected alternative. PAV even satisfies the stronger axiom *extended justified representation* (EJR). Representation axioms such as JR and EJR specify *what* proportionality guarantees a committee should satisfy. In this chapter, we focus instead on *stability*, which is strictly stronger than JR and EJR: An entitled coalition should not be able to switch to a proportional committee that gives *every* member strictly higher utility, even when the coalition's gains come from covering different approved alternatives rather than a single common one.

A standard way to formalize such coalitional stability under proportional entitlements is via the core for committee selection, introduced by Aziz et al. [ABC+17]. Intuitively, a coalition $\mathcal{V}' \subseteq \mathcal{V}$ is entitled to propose a committee of size proportional to its share. A size-k committee $W \subseteq \mathcal{C}$ is core-stable if there is no coalition $\mathcal{V}'$ and no committee W' with $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ such that every voter in $\mathcal{V}' \subseteq \mathcal{V}$ strictly improves her AV-score[1]

[1] Aziz et al. [ABC+17] defined the core using PAV-scores; as discussed in Observation 5.18, the

when moving from W to W'. This yields a cooperative-game interpretation with non-transferable utilities (NTU), where proportional entitlements restrict feasible deviations. A basic open question posed by Aziz et al. [ABC+17] is whether this core is always non-empty. This question was recently answered by Becker et al. [BGP26] who showed that every approval-based committee election admits a core-stable committee, and that one can be computed in polynomial time.

In this chapter, we generalize this committee-core viewpoint by defining *multi-winner games* (MW games) under both non-transferable (NTU) and transferable utilities (TU). The framework applies to multi-winner voting with arbitrary score-based preferences. In this chapter, we focus on *approval* preferences, i.e., multi-winner approval voting (see Section 2.6). The voters $\mathcal{V}$ are the players, and a coalition $\mathcal{V}' \subseteq \mathcal{V}$ may claim any *admissible* committee $W' \subseteq \mathcal{C}$ with $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$. Fixing a score-based voting rule, each admissible committee induces voter utilities according to the rule, and we study the resulting cooperative-game core as a set of *core-stable* utility functions.

Under the TU model, only a coalition's *total* induced utility matters, and it may be redistributed arbitrarily among its members. A utility function is TU-core-stable if it is induced by some size-k committee and no coalition $\mathcal{V}'$ can block it, i.e., $\mathcal{V}'$ should not be able to claim an admissible committee W' whose induced *total* score for $\mathcal{V}'$ exceeds the sum of utilities assigned to the voters in $\mathcal{V}'$.

Under the NTU model, utilities are not transferable: A coalition can only realize per-voter utilities that arise directly from some admissible committee. A utility function is *feasible* if it equals the per-voter score vector induced by some size-k committee. The NTU-core consists of all feasible utility functions that are not blocked by any coalition $\mathcal{V}'$, i.e., no coalition $\mathcal{V}' \subseteq \mathcal{V}$ and admissible committee W' exist such that every voter in $\mathcal{V}'$ obtains a strictly larger induced score from W' than her utility. Since every size-k committee induces a feasible utility function, the NTU-core under AV and SAV coincides with the committee core of Aziz et al. [ABC+17], which was defined via the PAV-score. The NTU-setting naturally models voters' utilities, with voters preferring better outcomes for themselves; whether it is the right model, however, depends on what a voter's utility stands for.

In political elections, it stands for satisfaction with the elected body and is generally not a divisible good. There is no mechanism by which one voter can be made better off by transferring another voter's satisfaction to her. Such situations can be described using the NTU model. However, in many environments, proportional 'entitlements' are not rigid per-voter guarantees but coalition-level resources that can be reallocated internally after the outcome is chosen. Interpreting utility as a proxy for surplus (e.g., influence, access, funding, or agenda benefits) makes TU natural: A coalition that can coordinate on a deviation can also write side agreements that compensate members who would otherwise lose from the deviation. We can therefore model a different group of settings: settings

resulting core coincides with the AV-score (and SAV-score) core. We use AV-scores because they are standard in subsequent work and simpler to compute.

where entities can contract with one another, and in which the decision is accompanied by a divisible resource that arrives with it. In these settings, "utility" can be interpreted as a bookkeeping device for a share of that resource rather than a measure of private satisfaction, and a transfer is an ordinary side agreement:

- **Budget and funding committees**: Consider a setting where a grant panel or a faculty committee selects a fixed number of projects to fund. The resource that is divided here is a share of the funding, the slots, or the staff time that follows from them. A transfer is a reallocation of that share. This can happen either directly, by one committee member offering resources to another (by moving funding, or by letting staff from its own group work on the other's project), or indirectly, by offering to support the other member's projects in a later round.
- **Shareholder and committee voting with side payments**: In corporate boards and similar bodies, each member casts one vote, and utility transfers (compensation, buyouts, monetary agreements between members) can occur. Here each chosen alternative provides monetary value that the members can then divide, so the divided resource is monetary.
- **Coalition formation in politics**: When parties form a government through a coalition, the ministries are divisible and tradeable. For example, even if the minister is of one party, state secretaries can belong to other parties, thereby "dividing" the ministry. In this case, the divided resource consists of the portfolios and the posts that come with them.
- **Participatory budgeting with transfers**: Voters express preferences over which projects to fund from a shared budget. In order to find mutually beneficial outcomes, neighborhood boards may trade budget shares across rounds or across categories. In this case, the divided resource is the budget share itself.

In each case the transferred object is the resource the decision distributes and not the voters' preferences over the committee. In this interpretation, TU-core stability is a no-profitable-renegotiation condition subject to proportional seat caps.

To illustrate this distinction, we give a small example where winning committees need not be core-stable, and allowing transfers (TU) can restore stability.

Example 5.1. *Let* $\mathcal{C} = \{c_1, c_2, c_3, c_4, c_5\}$ *and* $\mathcal{V} = \{v_1, \ldots, v_6\}$ *with* $\mathsf{A}_1 = \mathsf{A}_2 = \mathsf{A}_3 = \mathsf{A}_4 = \{c_1, c_2, c_3\}$ *and* $\mathsf{A}_5 = \mathsf{A}_6 = \{c_4, c_5\}$. *Recall that* A_i *is the set of approved alternatives for voter* v_i*, e.g.,* v_1 *approves the alternatives* c_1*,* c_2*, and* c_3*. Let* $k = 3$*. Then* $W = \{c_1, c_2, c_3\}$ *is the unique AV- and SAV-winning committee, with total AV-score* 12 *and SAV-score* 4*, inducing utilities* $(3, 3, 3, 3, 0, 0)$ *and* $(1, 1, 1, 1, 0, 0)$ *for AV and SAV, respectively. However, it is not core-stable (in the* NTU *sense): The coalition* $\{v_5, v_6\}$ *can propose* $\{c_4\}$ *(or* $\{c_5\}$*) and strictly improve both members. Thus every core-stable committee must contain* c_4 *or* c_5 *and two of* $\{c_1, c_2, c_3\}$*, inducing* $(2, 2, 2, 2, 1, 1)$ *and* $(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{1}{2}, \frac{1}{2})$ *for*

AV and SAV, respectively. In this instance, no AV-winning or SAV-winning committee is core-stable (under NTU). This phenomenon also occurs for PAV; see, e.g., Aziz et al. [ABC+17, Section 5.2]. For CC, any size-3 committee containing at least one of $\{c_4, c_5\}$ *(and hence at least one of* $\{c_1, c_2, c_3\}$*) is winning and induces an all-1 vector, which is certainly in the core. In fact, we will show that every JR-satisfying committee induces a core-stable utility function; see Proposition 5.20.*

In the TU model, an equal split of the maximum AV-score 12 *yields* $(2, 2, 2, 2, 2, 2)$*. This lies in the TU-core for AV, as every coalition* $\mathcal{V}' \neq \mathcal{V}$ *can achieve at most* $2|\mathcal{V}'|$ *utility and* $\mathcal{V}$ *can achieve at most* 12 *utility. Similarly, an equal split of the overall SAV-score* 4 *yields* $(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3})$*, which lies in the TU-core for SAV, as no coalition* $\mathcal{V}'$ *can exceed the sum of their utilities through a proportional sized committee. We will show how to compute the utility functions for AV and SAV in Algorithm 5.1. For PAV, a TU-core utility function assigns* 1.5 *to the first four voters and* 1 *to the remaining two. For CC, assigning* 1 *to every voter yields a TU-core utility function.*

Main contributions. We study the core of MW games under both NTU and TU for four prominent score-based rules: AV, SAV, CC, and PAV. Our results compare core *existence*, *structure*, and *computational complexity* across the two models and four rules.

Structural and constructive results.

1. **TU-AV/SAV:** We prove that the TU-core is always non-empty and give a polynomial-time algorithm (cf. Algorithm 5.1) to compute a core utility function for a broad class of voting rules in which each alternative contributes independently to each voter's score, and the approach generalizes even to rules over linear preferences such as the multi-winner Borda rule. The algorithm exploits the additivity of these rules by decomposing the total score into a coalition-independent baseline tied to the weakest winning alternative and an alternative-specific surplus that is payable only by voters who gain utility from these alternatives. We also provide an alternative non-emptiness proof via the Bondareva–Shapley theorem [Bon63, Sha67].

2. **TU-CC/PAV:** We show that the TU-core can be empty for CC and PAV. This shows that transferability alone does not guarantee stability. Additionally, for each $\lambda \in \{\text{AV}, \text{SAV}, \text{PAV}, \text{CC}\}$, we show that the Shapley value need not yield an element in the TU-core, even when the TU-core is non-empty.

3. **NTU-CC:** We show that the NTU-core under CC is "equivalent" to justified representation (JR). Combining this equivalence with a known greedy JR procedure (Algorithm 5.2) yields a polynomial-time method to compute an NTU-core utility function.

4. **NTU-AV/SAV/PAV:** We make explicit a simple correspondence between the NTU-cores induced by AV, SAV, and PAV (cf. Observation 5.18).

Table 5.1: Overview of structural and computational results. Each cell reports whether the core is guaranteed to be non-empty and the complexity of determining the core-nonemptiness and verifying whether a given utility function lies in the core, respectively. Constructive polynomial-time algorithms exist for TU-AV, TU-SAV (Algorithm 5.1), and for NTU constructive polynomial-time algorithms exist for all our studied rules (Becker et al. [BGP26] and Algorithm 5.2, via the equivalence with JR). Results in green are tractable (P = constructive in polynomial time), results in red are intractable ("-c" = complete, "-h" = hard). The upper bounds DP, Θ_2^{P}, and Δ_2^{P} follow from Theorem 5.24. References [T x] and [P x] point to Theorem and Proposition x of this thesis.

Rule	TU-Core Existence	TU-Core Verification	NTU-Core Existence	NTU-Core Verification
AV/SAV	Always (✓) [T5.10] P [T5.13]	coNP-c [T5.27/T5.31]	Always (✓) [BGP26] P [BGP26]	DP-c [T5.40]
CC	Can fail (✗) [P5.16] NP-h, in Θ_2^{P} [T5.26]	DP-c [T5.36]	Always (✓) [T5.22] P [T5.22]	NP-c [T5.43]
PAV	Can fail (✗) [P5.16] NP-h, in Δ_2^{P} [T5.26]	NP-h, in DP [P5.38]	Always (✓) [BGP26] P [BGP26]	DP-c [T5.40]

Taken together, these results show that the two axes of our framework—TU versus NTU and the choice of scoring rule—interact non-trivially: Under NTU, the core is non-empty for all four rules, whereas under TU it can be empty for CC and PAV. Verification behaves differently again: For CC it is harder under TU than under NTU, while for AV and SAV the opposite holds. The full 2×4 case distinction is thus necessary to map the stability landscape. Moreover, the pattern is not arbitrary: Under TU, feasibility ties core elements to winning committees, so rule-specific optimization structure directly governs stability—AV and SAV's separability yields universal existence, while the non-separability of CC and PAV permits emptiness. Indeed, the same mechanism as for AV and SAV extends to any totally separable rule (cf. Definition 5.8)—including rules over linear preferences. Under NTU, core-stable outcomes need not arise from winning committees (cf. Example 5.1), decoupling stability from the rule. Our positive NTU result—for CC—relies on an equivalence with JR, a property defined independently of the scoring rules. This suggests that the TU model is the natural setting in which the choice of voting rule substantively shapes coalitional stability.

Computational results. Similarly to Chapter 3, we study Core-Stability Existence (CE) and Core-Stability Verification (CV) for both TU and NTU-games, and establish tight complexity classifications in most cases (summarized in Table 5.1).

Related work. For background on cooperative games and the core, see Section 2.4. Our setting is an algorithmic cooperative game in which feasible deviations are constrained by

coalition-specific proportional entitlements, and utilities are induced by a multi-winner scoring rule.

A broad line of work on algorithmic cooperative games studies *combinatorial optimization games*, as seen in Chapter 3. Our TU model is similar in spirit in that coalition power is captured by an optimization objective, but differs in the feasibility constraint (committees bounded by $\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ rather than substructures) and in the utility source (scores induced by multi-winner rules rather than a fixed combinatorial objective). *Weighted voting games* [CEW11] constitute another classical family of algorithmic cooperative games. In a weighted voting game, each player carries a weight, and a coalition wins—and is worth 1—exactly when the total weight of its members meets or exceeds a fixed quota; otherwise it is worth 0. The core of the induced TU-game admits a simple characterization: It is non-empty if and only if the game has a veto player—a player contained in every winning coalition—in which case the core consists precisely of the imputations that split the unit value among the veto players [EGGW09]. Therefore, the non-emptiness of the core is decidable in polynomial time. Even though our MW games are also cooperative games based on voting, there are crucial differences. While in weighted voting games a coalition's worth is binary—either it is strong enough or it is not—in MW games a coalition's value is the maximum score that an admissible committee can achieve. Secondly, in MW games voters are not weighted and their ability to deviate only stems from the best committee that fulfills the proportional seat cap $\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$. Finally, core-stability existence in MW games can be intractable, depending on the underlying voting rule.

The concept of core-stable committees under the PAV rule was introduced by Aziz et al. [ABC+17], who interpreted proportional entitlements as constraints on deviations and posed non-emptiness as a basic open problem. This stability concept corresponds to our NTU-core under PAV, equivalently, under AV or SAV; see Observation 5.18. Subsequent work [JMW20, PS22, LS23, BGP+24, Pet25] defined the core with respect to AV since the two resulting core concepts coincide. Recently, Becker et al. [BGP26] showed that the core is always non-empty.

While the existence question was open, a natural approach was to relax the core: a committee is *c-approximately stable* if no coalition of voters $\mathcal{V}' \subseteq \mathcal{V}$ can deviate to a committee $W' \subseteq \mathcal{C}$ that every voter in the coalition prefers and whose size satisfies $c\,|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$—that is, the coalition is a factor c more entitled than W' requires. For $c = 1$, this recovers exact core stability. Jiang et al. [JMW20] studied a more general setting, where committees have different weights[2] and there is a maximum weight for admissible committees. They show that all monotone preference structures admit a 32-approximately stable committee; if the alternatives are unweighted, as in our setting, then they admit a 16-approximately stable committee. A different relaxation was given by Cheng et al. [CJMW20], who guaranteed existence by moving to randomization, proving

[2] In our setting, each committee has a weight equal to the number of alternatives in the committee.

that a *stable lottery* (a distribution over committees that is core-stable in expectation) always exists.

A different approach to relaxing the core has been to restrict the viable instances to guarantee the existence of a core-stable committee:

- **Restricted preference domains**: Pierczynski and Skowron [PS22] proved that for several restricted preference domains—including voter-interval, candidate-interval, and 1D-Euclidean domains—the core is guaranteed to be non-empty. They also provide voting rules that find a core-stable committee.
- **Bounded committee size**: Peters [Pet25] showed, via computer-aided linear-programming search, that the core is non-empty whenever the committee size satisfies $k \leq 8$, for any number of voters and candidates. This extended a previous result by Cheng et al. [CJMW20], who proved non-emptiness for $k = 3$ by an explicit case analysis.
- **Few alternatives**: Peters [Pet25] established non-emptiness whenever there are at most 15 alternatives, while Berker et al. [BTC$^+$26] developed a mixed-integer-programming and automated-reasoning framework that decides core (non-)emptiness for a fixed number of alternatives *independently* of the number of voters.

The *auditing* problem has been studied in the core setting. It corresponds to our Core-Stability Verification problem. Formally, it fixes a proposed outcome and asks whether some coalition can block it—that is, whether the outcome lies in the (approximate) core. Munagala et al. [MSW22] study this verification problem in participatory budgeting and give algorithms certifying (approximate) core membership.

Charikar et al. [CLR$^+$25] studied a different stability notion called *Condorcet winning sets*—a set of alternatives such that there exists no alternative outside the winning committee that is preferred over every alternative in the committee by at least half the voters. They show that for committee size 6, a Condorcet winning set is guaranteed to exist.

This chapter's contribution is orthogonal to the previous approaches: Rather than restricting preferences or moving to randomized outcomes, we broaden the model by allowing both NTU and TU interpretations of the same underlying rule-induced utilities and analyze how existence and computational properties change across rules and utility models. In particular, we provide existence and algorithmic results for AV and SAV in the TU model and for CC in the NTU model. Recently, Brill et al. [BGP$^+$24] showed that it is coNP-complete to determine whether a given committee is core-stable under NTU-AV. Per Observation 5.18, this also holds for NTU-SAV and NTU-PAV. The problem we study, NTU-core membership, is harder, as in addition to determining if there exists a blocking coalition, we also need to determine whether there exists a committee that

realizes the scores. Determining NTU-core membership is beyond NP; more precisely it is DP-complete[3] (cf. Theorem 5.40).

Related core notions have also been studied in participatory budgeting (PB), which generalizes multi-winner selection by allocating a budget across projects. Fain et al. [FGM16] initiated the study of (approximate) core outcomes in continuous PB and related them to market equilibria, while discrete PB variants have been studied algorithmically and from an approximation perspective [FMS18, JMW20, MSW22, SN25]. Recent work showed that core non-emptiness can fail for broad classes of PB utilities [Mal25]. While PB and our setting share the key idea of proportional resource entitlements for coalitions, the feasible outcomes and utility aggregation differ: PB allocates divisible (or budgeted) resources, whereas we study discrete committees and utilities induced by multi-winner scoring rules. Beyond cooperative deviations by voter coalitions, other game-theoretic models for multi-winner selection treat strategic behavior of candidates or projects. In the *multi-winner candidacy games* of Obraztsova et al. [OPEG20], the candidates are the players: Each candidate has preferences over which committees are elected and may withdraw in order for a beneficial committee to be selected. In the *project submission games* of Faliszewski et al. [FJK$^+$25], the strategic players are project proposers in participatory budgeting, each choosing which subset of its projects to submit so as to maximize the total cost of those that ultimately get funded. Both models study the strategic *entry or withdrawal* by candidates or proposers in non-cooperative games. Haret et al. [HKMS24] proposed a non-cooperative budgeting game in which outcomes arise from budget-allocation dynamics and proved core non-emptiness for restricted (tree-structured) domains. In contrast, we directly formulate TU and NTU cooperative games over committees, establishing, among others, non-emptiness of the TU-AV/SAV-core and NTU-CC-core for unrestricted domains.

Chapter outline. In Section 5.2, we introduce our MW games in TU and NTU and their related computational problems. In Section 5.3, we present our main structural results. In Section 5.4, we establish complexity classifications for Core-Stability Existence and Core-Stability Verification. We conclude in Section 5.5 with future research directions.

5.2 Preliminaries

For preliminaries on NTU and TU games, see Section 2.4. All games in this chapter are maximization games (cf. Definitions 2.45 and 2.50), as in our model, voters seek to maximize the utility. For preliminaries on multi-winner voting, see Section 2.6. For each voter coalition $\mathcal{V}' \subseteq \mathcal{V}$, we call a committee $W' \subseteq \mathcal{C}$ *share-admissible* (in short *admissible*) for $\mathcal{V}'$ if $|W'|$ does not exceed the proportional share of $\mathcal{V}'$, i.e., $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$. Then, let

$$\mathsf{sc}_\lambda(\mathcal{V}') \coloneqq \max_{W' \subseteq \mathcal{C} \,:\, |W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor} \mathsf{sc}_\lambda(\mathcal{V}', W') \tag{5.1}$$

[3] See Definition 2.17 for a definition.

denote the maximum score that an admissible committee can achieve for $\mathcal{V}'$. Moreover, let $\lambda(\mathcal{P}, k) = \arg \max_{W \subseteq \mathcal{C} \,:\, |W|=k} \mathsf{sc}_\lambda(\mathcal{V}, W)$ denote the set of all winning committees under rule λ.

5.2.1 Multi-Winner Games

Let λ denote a score-based voting rule[4]. A *λ-multi-winner game* consists of an MW-instance $(\mathcal{P}, k)$ with voter set $\mathcal{V}$ in $\mathcal{P}$, where the voters are the players. The utility of each voter coalition is derived from the maximum score that an admissible committee can achieve for it under λ. The goal is to find a *utility function* $\boldsymbol{\alpha} \colon \mathcal{V} \to \mathbb{R}_{\geq 0}$ (also called *utility vector*) that assigns to each voter v_i a utility value $\boldsymbol{\alpha}(v_i)$ such that no subset of voters has an incentive to deviate from the solution.

We propose two types of MW games: one with *transferable utilities* (TU) and the other with *non-transferable utilities* (NTU). In an NTU-game, a coalition can only realize utilities that arise from some committee (each voter receives her own score). In a TU-game, only the total coalition utility matters and can be redistributed arbitrarily among coalition members.

Definition 5.2 (MW game in TU and NTU)**.** *Fix a voting rule λ and an MW-instance $(\mathcal{P}, k)$; we define the corresponding TU- and NTU-games:*

The TU-game *induced by $(\mathcal{P}, k)$ and λ is $(\mathcal{V}, \mathit{TU}_\lambda\text{-}\chi)$ where, for every coalition $\mathcal{V}' \subseteq \mathcal{V}$,*

$$\mathsf{TU}_\lambda\text{-}\chi(\mathcal{V}') \;\coloneqq\; \mathsf{sc}_\lambda(\mathcal{V}') = \max_{W' \subseteq \mathcal{C} \,:\, |W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor} \mathsf{sc}_\lambda(\mathcal{V}', W').$$

The NTU-game *induced by $(\mathcal{P}, k)$ and λ is $(\mathcal{V}, \mathit{NTU}_\lambda\text{-}\mu(\mathcal{V}'))$, where*

$$\mathsf{NTU}_\lambda\text{-}\mu(\mathcal{V}') \coloneqq \Big\{ \boldsymbol{\alpha}' \colon \mathcal{V}' \to \mathbb{R}_{\geq 0} \;\Big|\; \exists W' \subseteq \mathcal{C} \text{ with } |W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor \text{ s.t. } \boldsymbol{\alpha}'(v) = \mathsf{sc}_\lambda(v, W') \;\forall v \in \mathcal{V}' \Big\}.$$

Blocking coalitions and the core follow the definitions given in Definitions 2.46 and 2.51. For the sake of completeness, we present the definitions for our games:

Definition 5.3 (Blocking, witness, and core)**.** *Let $(\mathcal{P}, k)$ be an MW-instance with $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$, $\boldsymbol{\alpha} \colon \mathcal{V} \to \mathbb{R}_{\geq 0}$ an allocation, and $\mathcal{V}' \subseteq \mathcal{V}$ a voter coalition.*

$\mathcal{V}'$ is said to be TU_λ-blocking $\boldsymbol{\alpha}$ if

$$\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) \;<\; \mathit{TU}_\lambda\text{-}\chi(\mathcal{V}'). \tag{5.2}$$

[4] In this chapter, we instantiate it only for approval preferences.

$\mathcal{V}'$ is said to be NTU_λ*-blocking* $\boldsymbol{\alpha}$ *if there exists a utility function* $\boldsymbol{\alpha}' \in \mathsf{NTU}_\lambda\text{-}\mu(\mathcal{V}')$ *such that*

$$\boldsymbol{\alpha}(v) < \boldsymbol{\alpha}'(v) \quad \text{for all } v \in \mathcal{V}'. \tag{5.3}$$

A blocking witness for $\mathcal{V}'$ (against $\boldsymbol{\alpha}$) is additional information that certifies that $\mathcal{V}'$ is blocking.

- *A committee W' is a witness for $\mathcal{V}'$ to be* TU_λ*-blocking for $\boldsymbol{\alpha}$ if $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ and $\sum_{v\in\mathcal{V}'} \boldsymbol{\alpha}(v) < \mathsf{sc}_\lambda(\mathcal{V}', W')$. Note that this implies that (5.2) holds.*
- *A committee W' is a witness for $\mathcal{V}'$ to be* NTU_λ*-blocking for $\boldsymbol{\alpha}$ if $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ and $\boldsymbol{\alpha}(v) < \mathsf{sc}_\lambda(v, W')$ for all $v \in \mathcal{V}'$. Note that this implies that (5.3) holds.*

We say that $\boldsymbol{\alpha}$ is TU_λ*-core-stable if $\sum_{v\in\mathcal{V}} \boldsymbol{\alpha}(v) = \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V})$, and no coalition is TU_λ-blocking for $\boldsymbol{\alpha}$. The condition $\sum_{v\in\mathcal{V}} \boldsymbol{\alpha}(v) = \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V})$ corresponds to $\boldsymbol{\alpha}$ being a preimputation. Due to the nature of $\boldsymbol{\alpha}$ as a distribution of utility, we will also call this condition an efficiency constraint. It is* NTU_λ*-core-stable if $\boldsymbol{\alpha} \in \mathsf{NTU}_\lambda\text{-}\mu(\mathcal{V})$, and no coalition is NTU_λ-blocking for $\boldsymbol{\alpha}$.*

The NTU_λ*-core (resp. TU_λ-core) is the set of all NTU_λ-core-stable (resp. TU_λ-core-stable) utility functions.*

Remark 5.4. *The NTU-game of Definition 5.2 should not be confused with the NTU representation of the TU-game (see Remark 2.52). In the TU-representation, any non-negative utility vector summing to at most $\mathsf{TU}_\lambda\text{-}\chi(\mathcal{V}')$ is feasible for a coalition $\mathcal{V}'$, whereas in the NTU-game, only utility vectors directly realized by some admissible committee are feasible. In particular, $\mathsf{NTU}_\lambda\text{-}\mu(\mathcal{V}') \subset \mathsf{TU}_\lambda\text{-}\mu(\mathcal{V}')$ in general, and core non-emptiness in one model does not imply it in the other.*

We illustrate the TU and NTU-cores on two small instances (with $k = 2$) for rules $\lambda \in \{\mathrm{AV}, \mathrm{SAV}, \mathrm{PAV}, \mathrm{CC}\}$. The following example shows the power of redistributing the utilities compared to non-transferable utilities:

Example 5.5 (TU game)**.** *Recall the instance from Example 2.59:*
Let $\mathcal{C} = \{c_1, c_2, c_3\}$, $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$, and

$$\mathsf{A}_1 = \mathsf{A}_2 = \{c_1\},\ \mathsf{A}_3 = \{c_1, c_2\},\ \mathsf{A}_4 = \{c_2, c_3\}$$

with $k = 2$. In the TU model, $\boldsymbol{\alpha}$ is core-stable if and only if it is feasible[5] *and satisfies the coalition inequalities*[6] *for all $\mathcal{V}' \subseteq \mathcal{V}$. Since singleton coalitions have seat cap 0, the binding constraints come from size-2 and -3 coalitions (seat cap 1). Reading off the best*

[5] As a reminder: $\sum_{i=1}^{4} \boldsymbol{\alpha}(v_i) = \max_{|W|=2} \mathsf{sc}_\lambda(\mathcal{V}, W)$.
[6] As a reminder: $\sum_{v_i\in\mathcal{V}'} \boldsymbol{\alpha}(v_i) \geq \max_{|W'|\leq\lfloor 2|\mathcal{V}'|/4\rfloor} \mathsf{sc}_\lambda(\mathcal{V}', W')$.

single-alternative score for each coalition gives exactly the inequalities below. For AV, CC, and PAV, these reduce to the same set of coalition inequalities on $(\boldsymbol{\alpha}(v_i))_{i\in[4]}$*; only the feasibility for the grand coalition differs. For SAV we get a separate set of inequalities. The coalition inequalities for AV, CC, and PAV are*

$$\begin{gathered}\boldsymbol{\alpha}(v_i) \geq 0, \text{ for all } i \in [4], \quad \boldsymbol{\alpha}(v_i) + \boldsymbol{\alpha}(v_j) \geq 2, \text{ for all } i \neq j \in [3],\\ \boldsymbol{\alpha}(v_1) + \boldsymbol{\alpha}(v_4) \geq 1, \quad \boldsymbol{\alpha}(v_2) + \boldsymbol{\alpha}(v_4) \geq 1, \quad \boldsymbol{\alpha}(v_3) + \boldsymbol{\alpha}(v_4) \geq 2\\ \boldsymbol{\alpha}(v_1) + \boldsymbol{\alpha}(v_2) + \boldsymbol{\alpha}(v_3) \geq 3.\end{gathered}$$

AV: $W_1 = \{c_1, c_2\}$ *is the unique winning committee with score* 5*, so* $\sum_{i\in[4]} \boldsymbol{\alpha}(v_i) = 5$*. By combining the coalition inequalities with the feasibility constraint, we get that the* TU_{AV}*-core is* $\boldsymbol{\alpha} = (1+\epsilon_1,\, 1+\epsilon_2,\, 1+\epsilon_3,\, 2-(\epsilon_1+\epsilon_2+\epsilon_3))$ *where* $0 \leq \epsilon_1+\epsilon_2+\epsilon_3 \leq 2$*,* $0 \leq \epsilon_1+\epsilon_2 \leq 1$*,* $0 \leq \epsilon_1+\epsilon_3 \leq 2$*,* $0 \leq \epsilon_2+\epsilon_3 \leq 2$*. Note that the vector notation is read as* $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}(v_i)$*. For example, by combining the two inequalities*

$$\begin{aligned}\boldsymbol{\alpha}(v_1) + \boldsymbol{\alpha}(v_2) + \boldsymbol{\alpha}(v_3) &= 3 + \epsilon_1 + \epsilon_2 + \epsilon_3 \geq 3\\ \text{and } \boldsymbol{\alpha}(v_4) &= 2 - (\epsilon_1 + \epsilon_2 + \epsilon_3) \geq 0\end{aligned}$$

we get that $0 \leq \epsilon_1+\epsilon_2+\epsilon_3 \leq 2$*. Similarly, all other inequalities can be derived.*

CC: The winning committees are W_1 *and* $W_2 = \{c_1, c_3\}$*, each with score* 4*, so* $\sum_{i\in[4]} \boldsymbol{\alpha}(v_i) = 4$*. By combining the coalition inequalities with the feasibility constraint, we get that the* TU_{CC}*-core is* $\boldsymbol{\alpha} = (1+\epsilon_1,\, 1+\epsilon_2,\, 1+\epsilon_3,\, 1-(\epsilon_1+\epsilon_2+\epsilon_3))$ *where* $0 \leq \epsilon_1+\epsilon_2+\epsilon_3 \leq 1$*,* $\epsilon_1+\epsilon_2 = 0$*,* $0 \leq \epsilon_1+\epsilon_3 \leq 1$*,* $0 \leq \epsilon_2+\epsilon_3 \leq 1$*.*

PAV: W_1 *is the unique winning committee with score* 4.5*, so* $\sum_{i\in[4]} \boldsymbol{\alpha}(v_i) = 4.5$*. By combining the coalition inequalities with the feasibility constraint, we get that the* TU_{PAV}*-core is* $\boldsymbol{\alpha} = (1+\epsilon_1,\, 1+\epsilon_2,\, 1+\epsilon_3,\, 1.5-(\epsilon_1+\epsilon_2+\epsilon_3))$ *where* $0 \leq \epsilon_1+\epsilon_2+\epsilon_3 \leq 1.5$*,* $0 \leq \epsilon_1+\epsilon_2 \leq 0.5$*,* $0 \leq \epsilon_1+\epsilon_3 \leq 1.5$*,* $0 \leq \epsilon_2+\epsilon_3 \leq 1.5$*.*

For SAV, we get the following coalition inequalities:

$$\begin{gathered}\boldsymbol{\alpha}(v_i) \geq 0, \text{ for all } i \in [4],\\ \boldsymbol{\alpha}(v_i) + \boldsymbol{\alpha}(v_3) \geq 1.5, \text{ for all } i \in [2],\\ \boldsymbol{\alpha}(v_i) + \boldsymbol{\alpha}(v_4) \geq 1, \text{ for all } i \in [3],\\ \boldsymbol{\alpha}(v_1) + \boldsymbol{\alpha}(v_2) \geq 2, \quad \boldsymbol{\alpha}(v_1) + \boldsymbol{\alpha}(v_2) + \boldsymbol{\alpha}(v_3) \geq 2.5.\end{gathered}$$

SAV: W_1 *is the unique winning committee with score* 3.5*, so* $\sum_{i\in[4]} \boldsymbol{\alpha}(v_i) = 3.5$*. By combining the coalition inequalities with the feasibility constraint, we get that the* TU_{SAV}*-core is* $\boldsymbol{\alpha} = (1+\epsilon_1,\, 1+\epsilon_2,\, 1+\epsilon_3,\, 0.5-(\epsilon_1+\epsilon_2+\epsilon_3))$ *where* $-0.5 \leq \epsilon_1+\epsilon_2+\epsilon_3 \leq 0.5$*,* $0 \leq \epsilon_1+\epsilon_2 \leq 0.5$*,* $-0.5 \leq \epsilon_1+\epsilon_3 \leq 0.5$*,* $-0.5 \leq \epsilon_2+\epsilon_3 \leq 0.5$*.*

Example 5.6 (NTU game)**.** *Let* $\mathcal{C} = \{c_1, c_2, c_3, c_4\}$*,* $\mathcal{V} = \{v_1, \ldots, v_6\}$*, and*

$$\mathsf{A}_1 = \{c_1\}, \quad \mathsf{A}_2 = \mathsf{A}_3 = \{c_1, c_4\}, \quad \mathsf{A}_4 = \{c_2\}, \quad \mathsf{A}_5 = \mathsf{A}_6 = \{c_2, c_3, c_4\}.$$

Let $k = 2$. Coalitions of size at most 2 have seat cap $\lfloor 2 \cdot 2/6 \rfloor = 0$ and cannot block. Coalitions of size 3 to 5 may select a single alternative. Let W be a committee of size $k = 2$ which yields a core-stable utility function. We analyze necessary and sufficient conditions on this committee W.

Necessary conditions. *If $W \cap \{c_1, c_4\} = \emptyset$, then v_1, v_2, v_3 each receive utility 0 and the coalition $\{v_1, v_2, v_3\}$ blocks via $\{c_1\}$. Symmetrically, $W \cap \{c_2, c_3, c_4\} = \emptyset$ lets $\{v_4, v_5, v_6\}$ block via $\{c_2\}$. Hence, every core-stable committee must intersect both $\{c_1, c_4\}$ and $\{c_2, c_3, c_4\}$.*

Sufficient conditions. *Suppose W meets both conditions. A coalition of size 3–5 deviating to $\{c_j\}$ requires every member to currently have utility 0 and to approve c_j. For c_1: Voters v_2, v_3 also approve c_4, so at most v_1 has utility 0. For c_2: Voters v_5, v_6 also approve c_3, c_4, so at most v_4 has utility 0. For c_3: Only two approvers exist. For c_4: Both conditions together ensure that at least four voters (v_2, v_3 via the first, v_5, v_6 via the second) already have utility ≥ 1. In every case, fewer than three zero-utility approvers exist. The grand coalition (with seat cap 2) cannot block either: strict improvement for v_1 requires $c_1 \notin W$ and for v_4 requires $c_2 \notin W$, as otherwise v_1 and v_4 cannot improve. This gives us $W = \{c_3, c_4\}$; but then voters v_5, v_6 who already have utility 2 are not strictly improved.*

Therefore the NTU_{AV}-core consists of the utility vectors induced by $\{c_1, c_2\}$, $\{c_1, c_3\}$, $\{c_1, c_4\}$, $\{c_2, c_4\}$, $\{c_3, c_4\}$. Since any blocking deviation selects a single alternative, the same five committees form the NTU-core under SAV, CC, and PAV.

5.2.2 Central Problems

In this chapter, we study two standard computational problems for cores: core non-emptiness and core membership verification. Let λ denote a score-based multi-winner voting rule.

λ-TU-CE (resp. λ-NTU-CE)
Input: An MW-instance $(\mathcal{P}, k)$.
Question: Is the TU_λ-core (resp. NTU_λ-core) of $(\mathcal{P}, k)$ non-empty?

λ-TU-CV (resp. λ-NTU-CV)
Input: An MW-instance $(\mathcal{P}, k)$ and a utility function $\boldsymbol{\alpha}$
Question: Is $\boldsymbol{\alpha}$ in the TU_λ-core (resp. NTU_λ-core) of $(\mathcal{P}, k)$?

5.3 Non-Emptiness and Structure of the Core

We now turn to our main structural results, organized by utility model.

5.3.1 TU: Existence and Construction

We start with the TU interpretation, where a coalition's induced total score can be redistributed arbitrarily among its members. Our first result shows that, under AV and SAV, the resulting TU-game always has a non-empty core. We are actually able to show an even stronger result: For any voting rule where the total committee score is the sum of individual voter scores and each alternative's contribution is independent of which others are selected, the resulting TU-game always has a non-empty core. We note that the TU-game under these voting rules is not convex since the TU characteristic function is not submodular (cf. Example 5.5 for a non-submodular instance), so core non-emptiness does not follow from standard sufficient conditions for convex games [Sha71]. Instead, we require a direct argument. We present two complementary perspectives. The Bondareva–Shapley argument [Bon63, Sha67] provides a conceptual certificate of non-emptiness via the fractional-committee relaxation, illuminating why the core is always non-empty: Linearity of these voting rules ensures that fractional committees cannot outperform integral ones. For these rules, direct combinatorial construction (Algorithm 5.1) additionally provides an explicit core element in polynomial time, which the Bondareva–Shapley proof alone does not guarantee. We then show that this positive picture is specific to AV and SAV: For CC and PAV the TU-core can be empty, and even when the core exists, canonical solution concepts such as the Shapley value need not produce a core element.

We next recall the Bondareva–Shapley Theorem [Bon63, Sha67].

Proposition 5.7 (Bondareva–Shapley Theorem[7] [Bon63, Sha67])**.** *Let $(\mathcal{V}, \chi)$ be a TU-game. The core of $(\mathcal{V}, \chi)$ is non-empty if and only if for every weight vector $\boldsymbol{\lambda} = (\lambda_S)_{S \subseteq \mathcal{V}}$ with $\lambda_S \geq 0$, $S \subseteq \mathcal{V}$, satisfying*

$$\sum_{S \subseteq \mathcal{V}:\, v_i \in S} \lambda_S = 1 \qquad \text{for all } v_i \in \mathcal{V}, \tag{5.4}$$

the following inequality holds: $$\sum_{S \subseteq \mathcal{V}} \lambda_S \, \chi(S) \leq \chi(\mathcal{V}). \tag{5.5}$$

In the following, we call a weight vector $\boldsymbol{\lambda}$ *balanced* if it is non-negative and satisfies (5.4). To show core non-emptiness, we will use Proposition 5.7 above and show that no balanced weight vector can collectively obtain more value than the grand coalition $\mathcal{V}$. The crucial idea is that every balanced vector yields a fractional committee, and the left-hand side of (5.5) can be upper-bounded by the optimum of a linear program over fractional committees. This linear program is maximized by selecting the top-k integral committee, which corresponds to a winning committee under the considered rules. First we define two terms that characterize the voting rules for which we show that the core is always non-empty.

[7] The Bondareva–Shapley Theorem is often stated with weights indexed only by non-empty coalitions, i.e., $\boldsymbol{\lambda} = (\lambda_S)_{\emptyset \neq S \subseteq \mathcal{V}}$. We include $\lambda_\emptyset$ purely for notational convenience (so we can write $\sum_{S \subseteq N}$), and this is without loss since $\nu(\emptyset) = 0$ in our setting and $\emptyset$ never appears in the balancing constraints (5.4).

Definition 5.8. *We say a score-based voting rule λ is* voter-separable *if* $\mathsf{sc}_\lambda(\mathcal{V}', W) = \sum_{v_i \in \mathcal{V}'} \mathsf{sc}_\lambda(v_i, W)$ *for all* $\mathcal{V}' \subseteq \mathcal{V}$ *and* $W \subseteq \mathcal{C}$. *We say a score-based voting rule λ is* alternative-separable *if* $\mathsf{sc}_\lambda(\mathcal{V}', W) = \sum_{c_i \in W} \mathsf{sc}_\lambda(\mathcal{V}', \{c_i\})$ *for all* $\mathcal{V}' \subseteq \mathcal{V}$ *and* $W \subseteq \mathcal{C}$. *We say a score-based voting rule λ is* totally separable *if it is voter-separable and alternative-separable.*

We can immediately observe that both AV and SAV are totally separable.

Observation 5.9. *AV and SAV are totally separable.*

Theorem 5.10. *TU_λ-core is always non-empty for all totally separable voting rules λ that satisfy $\mathsf{sc}_\lambda(v, \{c\}) \geq 0$ for all $v \in \mathcal{V}$ and $c \in \mathcal{C}$. Therefore, TU_{AV}-core and TU_{SAV}-core are always non-empty.*

Proof. Let $(\mathcal{P}, k)$ be an MW-instance with $\mathcal{P} = (\mathcal{C}, \mathcal{V})$. Let λ be a totally separable voting rule. For every voter coalition $S \subseteq \mathcal{V}$, let

$$\chi(S) \coloneqq \mathsf{TU}_\lambda\text{-}\chi(S) = \max_{W' \subseteq \mathcal{C}\colon |W'| \leq \lfloor \frac{|S|k}{|\mathcal{V}|} \rfloor} \mathsf{sc}_\lambda(S, W'), \text{ for } \lambda \in \{\mathrm{AV}, \mathrm{SAV}\}.$$

Note that for $\lambda = \mathrm{AV}$, $\chi(S) = \max_{W' \subseteq \mathcal{C}\colon |W'| \leq \lfloor \frac{|S|k}{|\mathcal{V}|} \rfloor} \sum_{v_i \in S} |\mathsf{A}_i \cap W'|$, and for $\lambda = \mathrm{SAV}$, $\chi(S) = \max_{W' \subseteq \mathcal{C}\colon |W'| \leq \lfloor \frac{|S|k}{|\mathcal{V}|} \rfloor} \sum_{v_i \in S} \frac{|\mathsf{A}_i \cap W'|}{|\mathsf{A}_i|}$. Then, by Proposition 5.7, it suffices to show that all *balanced* weight vectors satisfy inequality (5.5). Let $\boldsymbol{\lambda} = (\lambda_S)_{S \subseteq \mathcal{V}}$ be an arbitrary balanced weight vector satisfying (5.4). For each voter coalition $S \subseteq \mathcal{V}$, choose a committee W_S that maximizes the score for S, i.e.,

$$\chi(S) = \mathsf{sc}_\lambda(S, W_S) \text{ and } |W_S| \leq \lfloor \frac{|S|k}{|\mathcal{V}|} \rfloor. \tag{5.6}$$

Note that W_S depends on the voting rule, e.g., W_S can differ based on whether AV, SAV, or some other rule is the underlying voting rule. Then, we claim that the weighted sum of the sizes of the committees W_S is bounded by k.

Claim 5.11. $\sum_{S \subseteq \mathcal{V}} \lambda_S |W_S| \leq k$.

Proof. Using (5.4) we obtain the incidence-counting identity

$$\sum_{S \subseteq \mathcal{V}} \lambda_S |S| = \sum_{S \subseteq \mathcal{V}} \lambda_S \sum_{v_i \in S} 1 = \sum_{v_i \in \mathcal{V}} \sum_{S \subseteq \mathcal{V}\colon v_i \in S} \lambda_S \overset{(5.4)}{=} \sum_{v_i \in \mathcal{V}} 1 = |\mathcal{V}|. \tag{5.7}$$

Combining (5.6) with (5.7) gives

$$\sum_{S \subseteq \mathcal{V}} \lambda_S |W_S| \leq \sum_{S \subseteq \mathcal{V}} \lambda_S \lfloor \frac{k}{|\mathcal{V}|} |S| \rfloor \leq \frac{k}{|\mathcal{V}|} \sum_{S \subseteq \mathcal{V}} \lambda_S |S| = k \text{ as desired.}$$

(end of the proof of Claim 5.11 $\diamond$)

Next, we show that the weighted sum of the scores of the committees W_S is upper-bounded by the weighted sum of the scores of the associated fractional committee $\boldsymbol{x}$, where for each $c_j \in \mathcal{C}$, let $\boldsymbol{x}(c_j) \coloneqq \min\Big\{1, \sum_{S\subseteq\mathcal{V}\colon c_j\in W_S} \lambda_S\Big\}$.

Claim 5.12. *1.* $\sum_{c_j\in\mathcal{C}} \boldsymbol{x}(c_j) \leq k$.

2. $\sum_{S\subseteq\mathcal{V}} \lambda_S \cdot \mathsf{sc}_\lambda(S, W_S) \leq \sum_{c_j\in\mathcal{C}} \boldsymbol{x}(c_j) \cdot \mathsf{sc}_\lambda(\mathcal{V}, \{c_j\})$, *where* $\boldsymbol{x}(c_j) \coloneqq \min\Big\{1, \sum_{S\subseteq\mathcal{V}\colon c_j\in W_S} \lambda_S\Big\}$.

Proof. The first statement means that $\boldsymbol{x}$ as defined above yields a fractional committee of size at most k. By the definition of $\boldsymbol{x}$ and Claim 5.11, we obtain

$$\sum_{c_j\in\mathcal{C}} \boldsymbol{x}(c_j) \overset{\text{def}}{\leq} \sum_{c_j\in\mathcal{C}} \sum_{\substack{S\subseteq\mathcal{V}\colon\\ c_j\in W_S}} \lambda_S = \sum_{S\subseteq\mathcal{V}} \sum_{c_j\in W_S} \lambda_S = \sum_{S\subseteq\mathcal{V}} \lambda_S |W_S| \overset{\text{Claim 5.11}}{\leq} k,$$

showing the first statement. Then, by the definition of the score under totally separable voting rules and by counting the summation differently, we obtain

$$\sum_{S\subseteq\mathcal{V}} \lambda_S \cdot \mathsf{sc}_\lambda(S, W_S) = \sum_{S\subseteq\mathcal{V}} \lambda_S \sum_{v_i\in S} \sum_{c_j\in W_S} \mathsf{sc}_\lambda(v_i, \{c_j\}) = \sum_{c_j\in\mathcal{C}} \sum_{v_i\in\mathcal{V}} \mathsf{sc}_\lambda(v_i, \{c_j\}) \sum_{\substack{S\subseteq\mathcal{V}\colon\\ v_i\in S\wedge c_j\in W_S}} \lambda_S. \tag{5.8}$$

For each alternative $c_j \in \mathcal{C}$ and voter $v_i \in \mathcal{V}$, we infer the following.

$\sum_{\substack{S\subseteq\mathcal{V}\colon\\ v_i\in S\wedge c_j\in W_S}} \lambda_S \leq \sum_{S\subseteq\mathcal{V}\colon v_i\in S} \lambda_S \overset{(5.4)}{=} 1$, and $\sum_{\substack{S\subseteq\mathcal{V}\colon\\ v_i\in S\wedge c_j\in W_S}} \lambda_S \leq \sum_{\substack{S\subseteq\mathcal{V}\colon\\ c_j\in W_S}} \lambda_S$. Hence, $\sum_{\substack{S\subseteq\mathcal{V}\colon\\ v_i\in S\wedge c_j\in W_S}} \lambda_S \leq \min(1, \sum_{S\subseteq\mathcal{V}\colon c_j\in W_S} \lambda_S)$.

Plugging the inequality into (5.8), we obtain

$$\begin{aligned}\sum_{S\subseteq\mathcal{V}} \lambda_S \cdot \mathsf{sc}_\lambda(S, W_S) &\leq \sum_{c_j\in\mathcal{C}} \sum_{v_i\in\mathcal{V}} \mathsf{sc}_\lambda(v_i, \{c_j\}) \min(1, \sum_{S\subseteq\mathcal{V}\colon c_j\in W_S} \lambda_S) \\ &= \sum_{c_j\in\mathcal{C}} \sum_{v_i\in\mathcal{V}} \mathsf{sc}_\lambda(v_i, \{c_j\}) \boldsymbol{x}(c_j) = \sum_{c_j\in\mathcal{C}} \boldsymbol{x}(c_j) \cdot \mathsf{sc}_\lambda(\mathcal{V}, \{c_j\})\end{aligned}$$

(end of the proof of Claim 5.12 $\diamond$)

Finally, consider the following linear program

$$\max\Big\{\textstyle\sum_{c_j\in\mathcal{C}} y(c_j) \cdot \mathsf{sc}_\lambda(\mathcal{V}, \{c_j\})\colon 0 \leq y(c_j) \leq 1, \forall c_j \in \mathcal{C} \wedge \sum_{c_j\in\mathcal{C}} y(c_j) \leq k\Big\}.$$

Since $\mathsf{sc}_\lambda(\mathcal{V}, \{c_j\}) \geq 0$ holds for all $c_j \in \mathcal{C}$, the linear program is maximized by setting $y(c_j) = 1$ for the k alternatives with the largest score $\mathsf{sc}_\lambda(\mathcal{V}, \{c_j\})$ and $y(c_j) = 0$ otherwise. Since $\boldsymbol{x}$ is a feasible solution of the linear program (see Claim 5.12(1)), we infer

$$\sum_{c_j \in \mathcal{C}} \boldsymbol{x}(c_j) \cdot \mathsf{sc}_\lambda(\mathcal{V}, \{c_j\}) \leq \max_{W \subseteq \mathcal{C} \colon |W|=k} \sum_{c_j \in W} \mathsf{sc}_\lambda(\mathcal{V}, \{c_j\}) = \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V}).$$

Combining the above with Claim 5.12.2 gives

$$\sum_{S \subseteq \mathcal{V}} \lambda_S \chi(S) = \sum_{S \subseteq \mathcal{V}} \lambda_S \cdot \mathsf{sc}_\lambda(S, W_S) \leq \sum_{c_j \in \mathcal{C}} \boldsymbol{x}(c_j) \mathsf{sc}_\lambda(\mathcal{V}, \{c_j\}) \leq \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V}).$$

Thus (5.5) holds for every balanced weight vector, and Proposition 5.7 implies TU_λ-core $\neq \emptyset$. □

The LP-based algorithm derived from the core non-emptiness proof of Theorem 5.10 may run in exponential time. Next, we explicitly give a utility function in polynomial time and show that it is in the core; see Algorithm 5.1. Despite its compact form, our algorithm must satisfy core constraints for all coalitions and all admissible deviations. The baseline-plus-surplus split is designed so that any coalition's attainable score under its seat cap is upper-bounded by the utilities it is assigned. More precisely, our algorithm exploits the linearity of totally separable voting rules: Starting from a winning committee W, we distribute

1. a common baseline based on the weakest alternative in W to all voters (see line 5), and
2. the remaining surplus of each alternative only among its approving voters (see lines 2–3);

we prove that no coalition can block the resulting utility function.

Intuitively, the k-th best alternative c_k serves as a "price unit": Any coalition with seat cap s can extract at most $s|\mathsf{V}(c_k)|$ baseline approvals. Hence, any improvement beyond this baseline must come from selecting alternatives that beat c_k, whose excess approvals we pay only to voters who can actually realize them (the approvers).

Theorem 5.13. *Algorithm 5.1 computes a TU_λ-core-stable utility function in polynomial time for every totally separable voting rule λ that satisfies $\mathsf{sc}_\lambda(v, \{c\}) \geq 0$ for all $v \in \mathcal{V}$ and $c \in \mathcal{C}$.*

Proof. Let λ be a totally separable voting rule. Clearly, Algorithm 5.1 runs in polynomial time: Sorting alternatives by their λ-scores and computing the k utility functions can be done in polynomial time if the scores $\mathsf{sc}_\lambda(v, \{c\})$ are given for each voter-alternative pair. Let $c_1, \ldots, c_m$ be as sorted in Algorithm 5.1. By definition, $W = \{c_1, \ldots, c_k\}$ is a winning committee under λ. It remains to show that the returned function $\boldsymbol{\alpha}$ is in the TU-core.

We start by showing that the utility function is feasible for the grand coalition, i.e., $\sum_{v_i \in \mathcal{V}} \boldsymbol{\alpha}(v_i) = \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V}) = \sum_{j \in [k]} \mathsf{sc}_\lambda(\mathcal{V}, \{c_j\})$:

$$\sum_{v_i \in \mathcal{V}} \boldsymbol{\alpha}(v_i) = \sum_{j \in [k]} \sum_{v_i \in \mathcal{V}} \boldsymbol{\alpha}_j(v_i)$$

Algorithm 5.1: Compute a TU_λ-core-stable utility function for totally separable λ.

Input: A profile $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$ with $\mathsf{sc}_\lambda(v, \{c\})$ for all $v \in \mathcal{V}, c \in \mathcal{C}$ and $k \in [|\mathcal{C}|]$.

Sort and rename the alternatives such that $\mathsf{sc}_\lambda(\mathcal{V}, \{c_1\}) \geq \cdots \geq \mathsf{sc}_\lambda(\mathcal{V}, \{c_m\})$, breaking ties arbitrarily.

▷ Distribution of the surplus

foreach $(j, v_i) \in [k-1] \times \mathcal{V}$ **do**

  if $\mathsf{sc}_\lambda(v_i, \{c_j\}) \neq 0$ **then** $\boldsymbol{\alpha}_j(v_i) \leftarrow \frac{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\}) - \mathsf{sc}_\lambda(\mathcal{V},\{c_k\})}{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\})} \cdot \mathsf{sc}_\lambda(v_i, \{c_j\})$ **else** $\boldsymbol{\alpha}_j(v_i) \leftarrow 0$;

end

▷ Distribution of the baseline score

foreach $v_i \in \mathcal{V}$ **do** $\boldsymbol{\alpha}_k(v_i) \leftarrow \frac{k\,\mathsf{sc}_\lambda(\mathcal{V},\{c_k\})}{|\mathcal{V}|}$;

return $\sum_{j\in[k]} \boldsymbol{\alpha}_j$

$$= \sum_{j\in[k-1]} \sum_{\substack{v_i: \\ \mathsf{sc}_\lambda(v_i,\{c_j\})>0}} \frac{\mathsf{sc}_\lambda(\mathcal{V}, \{c_j\}) - \mathsf{sc}_\lambda(\mathcal{V}, \{c_k\})}{\mathsf{sc}_\lambda(\mathcal{V}, \{c_j\})} \cdot \mathsf{sc}_\lambda(v_i, \{c_j\}) \; + \sum_{v_i \in \mathcal{V}} \frac{k\,\mathsf{sc}_\lambda(\mathcal{V}, \{c_k\})}{|\mathcal{V}|}$$

$$= \sum_{j\in[k-1]} (\mathsf{sc}_\lambda(\mathcal{V}, \{c_j\}) - \mathsf{sc}_\lambda(\mathcal{V}, \{c_k\})) \; + \; k\mathsf{sc}_\lambda(\mathcal{V}, \{c_k\}) = \sum_{j\in[k]} \mathsf{sc}_\lambda(\mathcal{V}, \{c_j\}),$$

as desired. Now, we show that no coalition can block $\boldsymbol{\alpha}$. Fix an arbitrary coalition $\mathcal{V}' \subseteq \mathcal{V}$ and fix an admissible committee $W' \subseteq \mathcal{C}$ for $\mathcal{V}'$ with $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$. We will show that $\mathsf{sc}_\lambda(\mathcal{V}', W') \leq \sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i)$, which implies that $\mathcal{V}'$ cannot block $\boldsymbol{\alpha}$. First, split the alternatives from W' into alternatives from $\{c_1, \ldots, c_{k-1}\}$ that induce a surplus and the rest, and define $W_1 = \{c_j \in \{c_1, \ldots, c_{k-1}\} \cap W' \mid \mathsf{sc}_\lambda(\mathcal{V}, \{c_j\}) > \mathsf{sc}_\lambda(\mathcal{V}, \{c_k\})\}$ and $W_2 = W' \setminus W_1$. Then

$$\begin{aligned} \mathsf{sc}_\lambda(\mathcal{V}', W') = \sum_{c_j \in W'} \mathsf{sc}_\lambda(\mathcal{V}', \{c_j\}) &= \sum_{c_j \in W_1} \mathsf{sc}_\lambda(\mathcal{V}', \{c_j\}) + \sum_{c \in W_2} \mathsf{sc}_\lambda(\mathcal{V}', \{c\}) \\ &\leq \sum_{c_j \in W_1} \mathsf{sc}_\lambda(\mathcal{V}', \{c_j\}) + |W_2| \cdot \mathsf{sc}_\lambda(\mathcal{V}, \{c_k\}), \end{aligned} \tag{5.9}$$

where the last inequality holds because $W = \{c_1, \ldots, c_k\}$ consists of the k alternatives with highest λ-scores, and hence $\mathsf{sc}_\lambda(\mathcal{V}, \{c\}) \leq \mathsf{sc}_\lambda(\mathcal{V}, \{c_k\})$ for every $c \notin W_1$; moreover, $\mathsf{sc}_\lambda(\mathcal{V}', \{c\}) \leq \mathsf{sc}_\lambda(\mathcal{V}, \{c\})$.

Next, we lower-bound $\sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i)$.

Claim 5.14. $\displaystyle\sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i) \; \geq \; \sum_{c_j \in W_1} \mathsf{sc}_\lambda(\mathcal{V}', \{c_j\}) + |W_2| \cdot \mathsf{sc}_\lambda(\mathcal{V}, \{c_k\})$

Proof. By definition of $\boldsymbol{\alpha}$,

$$\sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i) = \sum_{v_i \in \mathcal{V}'} \sum_{j \in [k]} \boldsymbol{\alpha}_j(v_i) \geq \sum_{c_j \in W_1} \sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}_j(v_i) + \sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}_k(v_i)$$

$$
\begin{aligned}
&= \sum_{c_j \in W_1} \left(\sum_{\substack{v_i \in \mathcal{V}': \\ \mathsf{sc}_\lambda(v_i,\{c_j\})>0}} \frac{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\}) - \mathsf{sc}_\lambda(\mathcal{V},\{c_k\})}{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\})} \cdot \mathsf{sc}_\lambda(v_i,\{c_j\}) \right) + \sum_{v_i \in \mathcal{V}'} \frac{k\,\mathsf{sc}_\lambda(\mathcal{V},\{c_k\})}{|\mathcal{V}|} \\
&= \sum_{c_j \in W_1} \left(\sum_{\substack{v_i \in \mathcal{V}': \\ \mathsf{sc}_\lambda(v_i,\{c_j\})>0}} \frac{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\}) - \mathsf{sc}_\lambda(\mathcal{V},\{c_k\})}{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\})} \cdot \mathsf{sc}_\lambda(v_i,\{c_j\}) \right) + \frac{|\mathcal{V}'|k\,\mathsf{sc}_\lambda(\mathcal{V},\{c_k\})}{|\mathcal{V}|} \\
&\geq \sum_{c_j \in W_1} \left(\sum_{\substack{v_i \in \mathcal{V}': \\ \mathsf{sc}_\lambda(v_i,\{c_j\})>0}} \frac{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\}) - \mathsf{sc}_\lambda(\mathcal{V},\{c_k\})}{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\})} \cdot \mathsf{sc}_\lambda(v_i,\{c_j\}) \right) + (|W_1| + |W_2|)\mathsf{sc}_\lambda(\mathcal{V},\{c_k\}) \\
&= \sum_{c_j \in W_1} \left(\mathsf{sc}_\lambda(\mathcal{V},\{c_k\}) + \sum_{\substack{v_i \in \mathcal{V}': \\ \mathsf{sc}_\lambda(v_i,\{c_j\})>0}} \frac{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\}) - \mathsf{sc}_\lambda(\mathcal{V},\{c_k\})}{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\})} \cdot \mathsf{sc}_\lambda(v_i,\{c_j\}) \right) \\
&\quad + |W_2|\mathsf{sc}_\lambda(\mathcal{V},\{c_k\}) \\
&= \sum_{c_j \in W_1} \left(\mathsf{sc}_\lambda(\mathcal{V},\{c_k\}) + \frac{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\}) - \mathsf{sc}_\lambda(\mathcal{V},\{c_k\})}{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\})} \cdot (\mathsf{sc}_\lambda(\mathcal{V},\{c_j\}) - \sum_{v_i \notin \mathcal{V}'} \mathsf{sc}_\lambda(v_i,\{c_j\})) \right) \\
&\quad + |W_2|\mathsf{sc}_\lambda(\mathcal{V},\{c_k\}) \\
&\geq \sum_{c_j \in W_1} \left(\mathsf{sc}_\lambda(\mathcal{V},\{c_j\}) - \frac{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\})}{\mathsf{sc}_\lambda(\mathcal{V},\{c_j\})} \cdot \sum_{v_i \notin \mathcal{V}'} \mathsf{sc}_\lambda(v_i,\{c_j\}) \right) + |W_2|\mathsf{sc}_\lambda(\mathcal{V},\{c_k\}) \\
&= \sum_{c_j \in W_1} \mathsf{sc}_\lambda(\mathcal{V}',\{c_j\}) + |W_2| \cdot \mathsf{sc}_\lambda(\mathcal{V},\{c_k\})
\end{aligned}
$$

The first inequality holds since $|W'| = |W_1| + |W_2| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$.

For the fifth line, we use $\mathsf{sc}_\lambda(\mathcal{V},\{c_j\}) = \sum_{v_i \in \mathcal{V}'} \mathsf{sc}_\lambda(v_i,\{c_j\}) + \sum_{v_i \notin \mathcal{V}'} \mathsf{sc}_\lambda(v_i,\{c_j\})$, i.e., the voter-separable property. In the sixth line, we use that $\mathsf{sc}_\lambda(\mathcal{V},\{c_k\}) \geq 0$.

(end of the proof of Claim 5.14 $\diamond$)

Combining (5.9) with Claim 5.14 yields $\sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i) \geq \mathsf{sc}_\lambda(\mathcal{V}', W')$, as desired. Since $\mathcal{V}' \subseteq \mathcal{V}$ was arbitrary, no coalition blocks $\boldsymbol{\alpha}$, hence $\boldsymbol{\alpha}$ lies in the TU-core. □

Example 5.15. *Recall the instance from Example 5.5 with $k = 2$. Recall that Algorithm 5.1 builds each voter's payoff from two parts: a coalition-independent baseline tied to the weakest winning alternative (line 5), paid to every voter, plus the surplus of each stronger winning alternative over that baseline (line 2), shared only among the voters who approve it. For AV, the winning committee is $W = \{c_1, c_2\}$ with $\mathsf{sc}_{AV}(\mathcal{V},\{c_1\}) = 3$ and $\mathsf{sc}_{AV}(\mathcal{V},\{c_2\}) = 2$. Algorithm 5.1 gives every voter $\boldsymbol{\alpha}_2(v_i) = 2 \cdot 2/4 = 1$. The surplus of c_1*

over c_2 (line 3) is shared equally among c_1's three approving voters $\boldsymbol{\alpha}_1(v_i) = (3-2)/3$ for all $i \in [3]$, while $\boldsymbol{\alpha}_1(v_4) = 0$ since she does not approve c_1. Summing the two components yields $\boldsymbol{\alpha} = (1+1/3, 1+1/3, 1+1/3, 1+0) = (4/3, 4/3, 4/3, 1)$. For SAV, the winning committee is $W = \{c_1, c_2\}$ with $\mathsf{sc}_{SAV}(\mathcal{V}, \{c_1\}) = 2.5$ and $\mathsf{sc}_{SAV}(\mathcal{V}, \{c_2\}) = 1$. Algorithm 5.1 gives every voter $\boldsymbol{\alpha}_2(v_i) = 2 \cdot 1/4 = 0.5$.

The surplus of c_1 over c_2 (line 3) is shared among c_1's three approving voters according to the score they assign to c_1; $\boldsymbol{\alpha}_1(v_1) = \boldsymbol{\alpha}_1(v_2) = (2.5-1)/2.5 \cdot 1 = 0.6$ and $\boldsymbol{\alpha}_1(v_3) = (2.5-1)/2.5 \cdot 0.5 = 0.3$, while $\boldsymbol{\alpha}_1(v_4) = 0$ since she does not approve c_1. Summing the two components yields $\boldsymbol{\alpha} = (0.5+0.6, 0.5+0.6, 0.5+0.3, 0.5) = (1.1, 1.1, 0.8, 0.5)$.

The preceding results rely on the additivity of totally separable voting rules, which allows us to compare any "deviating" committee to a baseline alternative and to distribute "surplus" only among approvers. Note that Algorithm 5.1 and Theorem 5.10 also work for voting rules over linear preferences that are totally separable. This includes the multi-winner variant of the Borda rule. For CC and PAV, the voter-level scoring functions introduce complementarities and nonlinearities that break this structure, and core stability may fail even with transfers.

Proposition 5.16. *TU_{CC}-core and TU_{PAV}-core can be empty.*

Proof. Consider the instance $I = (\mathcal{P}, k)$ with voters $\mathcal{V} = \{v_1, \ldots, v_6\}$, alternatives $\mathcal{C} = \{c_1, c_2, c_3, c_4\}$, and $k = 2$. The approval preferences of the voters are as follows: $\mathsf{A}_1 = \{c_1, c_2\}$, $\mathsf{A}_2 = \{c_1, c_4\}$, $\mathsf{A}_3 = \{c_1, c_3\}$, $\mathsf{A}_4 = \{c_2, c_3\}$, $\mathsf{A}_5 = \{c_2, c_4\}$, $\mathsf{A}_6 = \{c_3, c_4\}$.

One can verify that for any size-2 committee, the CC-score is at most five, while the PAV-score is at most 5.5, i.e., $\mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) = 5$ and $\mathsf{TU}_{\mathrm{PAV}}\text{-}\chi(\mathcal{V}) = 5.5$. However, every alternative is approved by exactly three voters. Since $|\mathcal{V}|/k = 3$, the corresponding three voters can claim one seat and obtain a total score of three (under CC as well as PAV). These give four linear inequalities:

$$\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 \geq 3, \quad \boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_4 + \boldsymbol{\alpha}_5 \geq 3, \quad \boldsymbol{\alpha}_3 + \boldsymbol{\alpha}_4 + \boldsymbol{\alpha}_6 \geq 3, \quad \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_5 + \boldsymbol{\alpha}_6 \geq 3.$$

Combining them yields $\sum_{i \in [6]} \boldsymbol{\alpha}_i \geq 6$, which is not feasible given that the grand coalition can only achieve a score of at most 5 (resp. 5.5). □

Even when the TU-core is non-empty, core stability is not guaranteed by standard fairness-driven solution concepts. The next proposition formalizes this point for our setting: The Shapley value can assign too little total payoff to some entitled coalition, enabling a blocking deviation.

Proposition 5.17. *The utility function defined by the Shapley value is not always in TU_λ-core for any $\lambda \in \{AV, SAV, CC, PAV\}$, even if the core is not empty.*

Proof. We first consider AV. Consider the following profile $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$ with $\mathcal{C} = \{c_1, \ldots, c_5\}$, $\mathcal{V} = \{v_1, \ldots, v_6\}$, and $\mathsf{A}_1 = \mathsf{A}_2 = \mathsf{A}_3 = \{c_1, c_2\}$, $\mathsf{A}_4 = \mathsf{A}_5 = \mathsf{A}_6 = \{c_3, c_4, c_5\}$. Let $k = 4$.

Characteristic function. Every alternative has three approving voters: $|\mathsf{V}(c_j)| = 3$, $j \in [5]$. Seat caps are $\lfloor |\mathcal{V}'| \cdot 4/6 \rfloor$, giving cap $0, 1, 2, 2, 3, 4$ for $|\mathcal{V}'| = 1, \ldots, 6$. By symmetry, coalition values depend only on the group composition (g_1, g_2) (number of voters from each group). The non-trivial values are:

(g_1, g_2)	$(2,0)$	$(1,1)$	$(0,2)$	$(3,0)$	$(2,1)$	$(1,2)$	$(0,3)$	$(3,1)$	$(2,2)$	$(1,3)$	$(3,2)$	$(2,3)$	$(3,3)$
χ	2	1	2	6	4	4	6	6	4	6	8	9	12

The grand coalition value is $\chi(3,3) = 12$ (e.g., $\{c_1, c_2, c_3, c_4\}$ gives $3 \cdot 2 + 3 \cdot 2 = 12$).

Shapley value. For voter v_i,

$$\phi_i = \sum_{\mathcal{V}' \subseteq \mathcal{V} \setminus \{v_i\}} \frac{|\mathcal{V}'|!(|\mathcal{V}| - |\mathcal{V}'| - 1)!}{|\mathcal{V}|!} \cdot (\chi(\mathcal{V}' \cup \{v_i\}) - \chi(\mathcal{V}')).$$

By symmetry $\phi_1 = \phi_2 = \phi_3$ and $\phi_4 = \phi_5 = \phi_6$, with $\phi_1 + \phi_4 = 4$.

Grouping v_1's marginal contributions by (a, b), the number of members of $S \subseteq \mathcal{V} \setminus \{v_1\}$ from each group ($a \in \{0, 1, 2\}$, $b \in \{0, 1, 2, 3\}$), with multiplicity $\binom{2}{a}\binom{3}{b}$ and Shapley weight $\frac{(a+b)!\,(5-a-b)!}{6!}$, the nine *non-zero* terms are from $(0,1)$, $(0,2)$, $(1,0)$, $(1,1)$, $(1,3)$, $(2,0)$, $(2,1)$, $(2,2)$, $(2,3)$. This yields Shapley value $\phi_1 = \frac{19}{10}$ (we omit the routine calculation).

Blocking. Coalition $\{v_1, v_2, v_3\}$ has seat cap 2 and picks $\{c_1, c_2\}$, achieving score 6, while $\phi_1 + \phi_2 + \phi_3 = \frac{57}{10} < 6$. The $\mathsf{TU}_{\mathrm{AV}}$-core is non-empty by Theorem 5.10; Algorithm 5.1 yields $\boldsymbol{\alpha} = (2, 2, 2, 2, 2, 2)$.

We next consider SAV. Consider the following profile $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$ with $\mathcal{C} = \{c_1, \ldots, c_6\}$, $\mathcal{V} = \{v_1, \ldots, v_4\}$, and $\mathsf{A}_1 = \{c_1\}$, $\mathsf{A}_2 = \mathsf{A}_3 = \mathsf{A}_4 = \{c_2, \ldots, c_6\}$. Let $k = 2$.

Characteristic function. Seat caps are $\lfloor |\mathcal{V}'| \cdot 2/4 \rfloor$, giving cap $0, 1, 1, 2$ for $|\mathcal{V}'| = 1, \ldots, 4$. By symmetry, coalition values depend only on the group composition (g_1, g_2) (number of voters from each group). The non-trivial values are:

(g_1, g_2)	$(1,1)$	$(0,2)$	$(1,2)$	$(0,3)$	$(1,3)$
χ	1	$\frac{2}{5}$	1	$\frac{3}{5}$	$\frac{8}{5}$

The grand coalition value is $\chi(1,3) = \frac{8}{5}$ (e.g., $\{c_1, c_2\}$ gives $1 + \frac{3}{5} = \frac{8}{5}$).

Shapley value. By symmetry $\phi_2 = \phi_3 = \phi_4$, with $\phi_1 + \phi_2 + \phi_3 + \phi_4 = \frac{8}{5}$.

Grouping v_1's marginal contributions by $(0, a)$, the number of members of $S \subseteq \mathcal{V} \setminus \{v_1\}$ from the second group ($a \in \{1, 2, 3\}$), with multiplicity $\binom{3}{a}$ and Shapley weight $\frac{(a)!\,(3-a)!}{4!}$, the three *non-zero* terms are from $(0,1)$, $(0,2)$, $(0,3)$. This yields Shapley value $\phi_1 = \frac{65}{100}$ (we omit the routine calculation), which implies $\phi_2 = \phi_3 = \phi_4 = \frac{95}{300}$.

Blocking. Coalition $\{v_1, v_2\}$ has seat cap 1 and picks $\{c_1\}$, achieving score 1, while $\phi_1 + \phi_2 = \frac{290}{300} < 1$. The $\mathsf{TU}_{\mathrm{SAV}}$-core is non-empty by Theorem 5.10; $\boldsymbol{\alpha} = (1, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$ is in the core.

We now consider CC. Consider the following profile $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$ with $\mathcal{C} = \{c_1, c_2, c_3, c_4\}$, $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$, and $\mathsf{A}_1 = \{c_1, c_2\}$, $\mathsf{A}_2 = \{c_1, c_3\}$, $\mathsf{A}_3 = \{c_2, c_3\}$, $\mathsf{A}_4 = \{c_4\}$. Let $k = 2$.

Characteristic function. Every alternative except for c_4 has two approving voters: $|\mathsf{V}(c_j)| = 2$, $j \in [3]$ and $|\mathsf{V}(c_4)| = 1$. Seat caps are $\lfloor |\mathcal{V}'| \cdot \frac{2}{4} \rfloor$, giving cap $0, 1, 1, 2$ for $|\mathcal{V}'| = 1, \ldots, 4$. By symmetry, coalition values depend only on the group composition (g_1, g_2), where g_1 is the number of voters from $\{v_1, v_2, v_3\}$ and g_2 is the number of voters from $\{v_4\}$. The non-trivial values are:

(g_1, g_2)	$(2,0)$	$(1,1)$	$(3,0)$	$(2,1)$	$(3,1)$
χ	2	1	2	2	3

The grand coalition value is $\chi(3,1) = 3$ (e.g., $\{c_1, c_2\}$ gives 3).

Shapley value. By symmetry $\phi_1 = \phi_2 = \phi_3$. Grouping v_1's marginal contributions by (a, b) the number of members of $S \subseteq \mathcal{V} \setminus \{v_1\}$ from each group ($a \in \{1, 2\}$, $b \in \{0, 1\}$), with multiplicity $\binom{2}{a}\binom{1}{b}$ and Shapley weight $\frac{(a+b)!(3-a-b)!}{4!}$, the four *non-zero* terms are from $(1,0), (0,1), (1,1), (2,1)$. This yields Shapley value $\phi_1 = \frac{5}{6}$ (we omit the routine calculation).

Blocking. Coalition $\{v_1, v_2\}$ has seat cap 1 and picks $\{c_1\}$, achieving score 2, while $\phi_1 + \phi_2 = \frac{10}{6} < 2$. The TU-CC-core is non-empty, as $\boldsymbol{\alpha} = (1, 1, 1, 0)$ is in the core.

Finally, we consider PAV: Consider the same profile, as for CC.

Characteristic function. The characteristic function is mostly identical to the CC-case with the small difference of $\chi(3,1) = 3.5$. This gives us the following non-trivial values:

(g_1, g_2)	$(2,0)$	$(1,1)$	$(3,0)$	$(2,1)$	$(3,1)$
χ	2	1	2	2	3.5

Shapley value. Using the same methods and arguments as for CC, we get the Shapley value $\phi_1 = \frac{23}{24}$.

Blocking. Coalition $\{v_1, v_2\}$ has seat cap 1 and picks $\{c_1\}$, achieving score 2, while $\phi_1 + \phi_2 = \frac{23}{12} < 2$. The TU-PAV-core is non-empty, as $\boldsymbol{\alpha} = (1, 1, 1, 0.5)$ is in the core. □

In summary, the totally separable rules are the rules among the four for which the TU-core is unconditionally non-empty; we next ask how this picture changes when transfers are removed.

5.3.2 NTU: Equivalences and the CC Core

We now turn to the NTU interpretation, which matches the original committee-core model [ABC+17] in which coalitions cannot transfer utility, and blocking requires a deviation that strictly improves each member's realized score. This makes the core sensitive to the feasible *utility profiles* induced by committees, rather than only to total attainable score. We first clarify the AV-SAV-PAV equivalence (Observation 5.18) and then focus on CC.

Observation 5.18. *Let $(\mathcal{P}, k)$ be an MW-instance with $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$. For each coalition $\mathcal{V}' \subseteq \mathcal{V}$, the function $f_{\mathcal{V}'}\colon \mathsf{NTU}_{\mathrm{AV}}\text{-}\mu(\mathcal{V}') \to \mathsf{NTU}_{\mathrm{PAV}}\text{-}\mu(\mathcal{V}')$ defined by $f_{\mathcal{V}'}(\boldsymbol{\alpha})(v) = \sum_{z \in [\boldsymbol{\alpha}(v)]} 1/z$, for all $v \in \mathcal{V}'$, where the sum is interpreted as 0 if $\boldsymbol{\alpha}(v) = 0$ is a bijection, and both $f_{\mathcal{V}'}$ and $f_{\mathcal{V}'}^{-1}$ are computable in polynomial time. For each coalition $\mathcal{V}' \subseteq \mathcal{V}$, the function $g_{\mathcal{V}'}\colon \mathsf{NTU}_{\mathrm{AV}}\text{-}\mu(\mathcal{V}') \to \mathsf{NTU}_{\mathrm{SAV}}\text{-}\mu(\mathcal{V}')$ defined by $g_{\mathcal{V}'}(\boldsymbol{\alpha})(v) = \begin{cases} \frac{\boldsymbol{\alpha}(v)}{|\mathsf{A}(v)|} & \mathsf{A}(v) \neq \emptyset \\ 0 & \text{else} \end{cases}$, is a bijection, and both $g_{\mathcal{V}'}$ and $g_{\mathcal{V}'}^{-1}$ are computable in polynomial time. Consequently, for each coalition $\mathcal{V}' \subseteq \mathcal{V}$, the function $h_{\mathcal{V}'}\colon \mathsf{NTU}_{\mathrm{SAV}}\text{-}\mu(\mathcal{V}') \to \mathsf{NTU}_{\mathrm{PAV}}\text{-}\mu(\mathcal{V}')$ defined by $h_{\mathcal{V}'} = f_{\mathcal{V}'} \circ g_{\mathcal{V}'}^{-1}$ is a bijection, and both $h_{\mathcal{V}'}$ and $h_{\mathcal{V}'}^{-1}$ are computable in polynomial time. Moreover, for every $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}\text{-}\mu(\mathcal{V})$ and every coalition $\mathcal{V}' \subseteq \mathcal{V}$,*

$\mathcal{V}'$ is NTU-AV-blocking for $\boldsymbol{\alpha}$ if and only if it is NTU-PAV-blocking for $f_{\mathcal{V}'}(\boldsymbol{\alpha})$

and

$\mathcal{V}'$ is NTU-AV-blocking for $\boldsymbol{\alpha}$ if and only if it is NTU-SAV-blocking for $g_{\mathcal{V}'}(\boldsymbol{\alpha})$.

As a consequence, for every $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{SAV}}\text{-}\mu(\mathcal{V})$ and every coalition $\mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{V}'$ is NTU-SAV-blocking for $\boldsymbol{\alpha}$ if and only if it is NTU-PAV-blocking for $h_{\mathcal{V}'}(\boldsymbol{\alpha})$.

Proof. We first show the statements regarding the connection between AV and PAV: To simplify the notation, for each fixed $x \in \{0, \ldots, k\}$ let $\phi(x) \coloneqq \sum_{z=1}^{x} \frac{1}{z}$ (with $\phi(0) = 0$) denote the harmonic sum over x. Note that $f_{\mathcal{V}'}(\boldsymbol{\alpha})(v) = \phi(\boldsymbol{\alpha}(v))$ for all $v \in \mathcal{V}'$.

We first show that f is a bijection. Let $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}\text{-}\mu(\mathcal{V}')$ be a feasible utility function for $\mathcal{V}'$ under AV. By feasibility, there exists a committee W of size k such that $\boldsymbol{\alpha}(v) = \mathsf{sc}_{\mathrm{AV}}(v, W)$ for all $v \in \mathcal{V}'$. By the definition of PAV,

$$\mathsf{sc}_{\mathrm{PAV}}(v, W) = \phi(\boldsymbol{\alpha}(v)) = f_{\mathcal{V}'}(\boldsymbol{\alpha})(v),$$

hence $f_{\mathcal{V}'}(\boldsymbol{\alpha}) \in \mathsf{NTU}_{\mathrm{PAV}}\text{-}\mu(\mathcal{V}')$.

Conversely, let $\boldsymbol{\alpha}' \in \mathsf{NTU}_{\mathrm{PAV}}\text{-}\mu(\mathcal{V}')$ and let W witness feasibility, i.e., $\boldsymbol{\alpha}'(v) = \mathsf{sc}_{\mathrm{PAV}}(v, W) = \phi(\mathsf{sc}_{\mathrm{AV}}(v, W))$ for all $v \in \mathcal{V}'$. Define $\boldsymbol{\alpha}(v) = \mathsf{sc}_{\mathrm{AV}}(v, W) \in \{0, \dots, k\}$. Then $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}\text{-}\mu(\mathcal{V}')$ and $f_{\mathcal{V}'}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}'$.

Since ϕ is strictly increasing on $\{0, \dots, k\}$, it is injective; therefore $f_{\mathcal{V}'}$ is a bijection. Polynomial-time computability follows by precomputing $\phi(0), \dots, \phi(k)$ and inverting via table lookup.

Now, we show the blocking equivalence. Let $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}\text{-}\mu(\mathcal{V}')$ and write $\boldsymbol{\alpha}' = f_{\mathcal{V}'}(\boldsymbol{\alpha})$. Assume that $\mathcal{V}'$ is $\mathsf{NTU}_{\mathrm{AV}}$-blocking $\boldsymbol{\alpha}$. By definition, let W' be an admissible committee with $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ such that for all $v \in \mathcal{V}'$, $\mathsf{sc}_{\mathrm{AV}}(v, W') > \boldsymbol{\alpha}(v)$. Applying ϕ and using strict monotonicity, for all $v \in \mathcal{V}'$ we obtain $\phi(\mathsf{sc}_{\mathrm{AV}}(v, W')) > \phi(\boldsymbol{\alpha}(v))$. Using $\mathsf{sc}_{\mathrm{PAV}}(v, W') = \phi(\mathsf{sc}_{\mathrm{AV}}(v, W'))$ and $\boldsymbol{\alpha}'(v) = \phi(\boldsymbol{\alpha}(v))$, this is equivalent to $\mathsf{sc}_{\mathrm{PAV}}(v, W') > \boldsymbol{\alpha}'(v)$ for all $v \in \mathcal{V}'$, hence $\mathcal{V}'$ is $\mathsf{NTU}_{\mathrm{PAV}}$-blocking $\boldsymbol{\alpha}'$.

The reverse implication is analogous (or follows by applying the same argument to $f_{\mathcal{V}'}^{-1}$). We now show the statements regarding the connection between AV and SAV: We first show that g is a bijection. Let $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}\text{-}\mu(\mathcal{V}')$ be a feasible utility function for $\mathcal{V}'$ under AV. By feasibility, there exists a committee W of size k such that $\boldsymbol{\alpha}(v) = \mathsf{sc}_{\mathrm{AV}}(v, W)$ for all $v \in \mathcal{V}'$. By the definition of SAV,

$$\mathsf{sc}_{\mathrm{SAV}}(v, W) = \begin{cases} \frac{|\mathsf{A}(v) \cap W|}{|\mathsf{A}(v)|} & \mathsf{A}(v) \neq \emptyset \\ 0 & \text{else} \end{cases} = g_{\mathcal{V}'}(\boldsymbol{\alpha})(v)$$

hence $g_{\mathcal{V}'}(\boldsymbol{\alpha}) \in \mathsf{NTU}_{\mathrm{SAV}}\text{-}\mu(\mathcal{V}')$.

Conversely, let $\boldsymbol{\alpha}' \in \mathsf{NTU}_{\mathrm{SAV}}\text{-}\mu(\mathcal{V}')$ and let W witness feasibility, i.e., $\boldsymbol{\alpha}'(v) = \mathsf{sc}_{\mathrm{SAV}}(v, W) = \begin{cases} \frac{|\mathsf{A}(v) \cap W|}{|\mathsf{A}(v)|} & \mathsf{A}(v) \neq \emptyset \\ 0 & \text{else} \end{cases}$ for all $v \in \mathcal{V}'$. Define $\boldsymbol{\alpha}(v) = \mathsf{sc}_{\mathrm{AV}}(v, W) \in \{0, \dots, k\}$. Then $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}\text{-}\mu(\mathcal{V}')$ and $g_{\mathcal{V}'}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}'$. Since the value of $\frac{|\mathsf{A}(v) \cap W|}{|\mathsf{A}(v)|}$ is strictly increasing with increasing $|\mathsf{A}(v) \cap W|$, it is injective; therefore $g_{\mathcal{V}'}$ is a bijection. Polynomial-time computability follows directly by definition of the scores, as we just divide (or multiply) by $|\mathsf{A}(v)|$.

Now, we show the blocking equivalence. Let $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}\text{-}\mu(\mathcal{V}')$ and write $\boldsymbol{\alpha}' = g_{\mathcal{V}'}(\boldsymbol{\alpha})$. Assume that $\mathcal{V}'$ is $\mathsf{NTU}_{\mathrm{AV}}$-blocking $\boldsymbol{\alpha}$. By definition, let W' be an admissible committee with $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ such that for all $v \in \mathcal{V}'$, $\mathsf{sc}_{\mathrm{AV}}(v, W') > \boldsymbol{\alpha}(v)$. As all voters must improve, we know that for all $v \in \mathcal{V}'$ $\mathsf{A}(v) \neq \emptyset$.

By dividing by $|\mathsf{A}(v)|$, for all $v \in \mathcal{V}'$ we obtain $\frac{\mathsf{sc}_{\mathrm{AV}}(v, W')}{|\mathsf{A}(v)|} > \frac{\boldsymbol{\alpha}(v)}{|\mathsf{A}(v)|}$. Using $\mathsf{sc}_{\mathrm{SAV}}(v, W') = \frac{\mathsf{sc}_{\mathrm{AV}}(v, W')}{|\mathsf{A}(v)|}$ and $\boldsymbol{\alpha}'(v) = \frac{\boldsymbol{\alpha}(v)}{|\mathsf{A}(v)|}$, this is equivalent to $\mathsf{sc}_{\mathrm{SAV}}(v, W') > \boldsymbol{\alpha}'(v)$ for all $v \in \mathcal{V}'$, hence $\mathcal{V}'$ is $\mathsf{NTU}_{\mathrm{SAV}}$-blocking $\boldsymbol{\alpha}'$. The reverse implication is analogous (or follows by applying the same argument to $g_{\mathcal{V}'}^{-1}$).

Finally, the statement for h and the blocking equivalence follows directly by chaining the two above arguments. □

Observation 5.18 shows that, in the NTU model, the AV-, SAV-, and PAV-induced cores are equivalent up to a monotone transformation of voter utilities. For CC, by contrast, we can resolve existence unconditionally by exploiting a direct correspondence between core stability and representation. For completeness, we recall the following definition.

Definition 5.19 (Justified Representation (JR) [ABC+17])**.** *Let $(\mathcal{P}, k)$ denote an MW-instance with $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$. A committee W with $|W| = k$ satisfies* justified representation *(JR) if for each cohesive voter coalition $\mathcal{V}' \subseteq \mathcal{V}$, it holds that $W \cap \bigcup_{v_i \in \mathcal{V}'} \mathsf{A}_i \neq \emptyset$. $\mathcal{V}'$ is* cohesive *if $|\mathcal{V}'| \geq \lceil \frac{|\mathcal{V}|}{k} \rceil$ and $\bigcap_{v_i \in \mathcal{V}'} \mathsf{A}_i \neq \emptyset$.*

For CC, feasibility and blocking simplify because each voter's utility is binary. A coalition can block only by moving from utility 0 to utility 1 for every member, i.e., by selecting a committee that "covers" each member with at least one approved alternative. Under proportional entitlements, this is closely aligned with the JR axiom. The next proposition makes this correspondence exact: Core stability is equivalent to JR of the witnessing committee.

Proposition 5.20. *Let $\boldsymbol{\alpha}$ be an NTU-utility function feasible for $\mathcal{V}$ under CC, and let W be a size-k committee witness, i.e., $\boldsymbol{\alpha}(v_i) = \mathsf{sc}_{CC}(v_i, W)$ holds for all $v_i \in \mathcal{V}$. Then, $\boldsymbol{\alpha} \in \mathsf{NTU}_{CC}$-core if and only if W satisfies JR.*

Proof. Let $\boldsymbol{\alpha}$ and W be as in the statement. Recall that for CC we have $\mathsf{sc}_{\mathrm{CC}}(v_i, W) \in \{0, 1\}$, and $\mathsf{sc}_{\mathrm{CC}}(v_i, W) = 1$ if and only if $W \cap \mathsf{A}_i \neq \emptyset$. To show the statement, we consider two cases: W satisfies JR and it does not.

Assume that W satisfies JR and suppose, for contradiction, that $\boldsymbol{\alpha} \notin \mathsf{NTU}_{\mathrm{CC}}$-core. Then there exist a voter coalition $\mathcal{V}' \subseteq \mathcal{V}$ and an admissible committee $W' \subseteq \mathcal{C}$ with $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ such that $\mathsf{sc}_{\mathrm{CC}}(v_i, W') > \boldsymbol{\alpha}(v_i)$ for all $v_i \in \mathcal{V}'$. Since both terms lie in $\{0, 1\}$, we have $\mathsf{sc}_{\mathrm{CC}}(v_i, W') = 1$ and $\boldsymbol{\alpha}(v_i) = 0$ for all $v_i \in \mathcal{V}'$. Hence each $v_i \in \mathcal{V}'$ approves at least one alternative in W', so by pigeonhole there exists $c \in W'$ approved by at least $\lceil \frac{|\mathcal{V}'|}{|W'|} \rceil$ voters in $\mathcal{V}'$. Let $\mathcal{V}'' \coloneqq \{v_i \in \mathcal{V}' \colon c \in \mathsf{A}_i\}$; then $|\mathcal{V}''| \geq \lceil \frac{|\mathcal{V}'|}{|W'|} \rceil \geq \lceil \frac{|\mathcal{V}|}{k} \rceil$, where the last inequality uses admissibility $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$. Thus $c \in \bigcap_{v_i \in \mathcal{V}''} \mathsf{A}_i$ and $|\mathcal{V}''| \geq \lceil \frac{|\mathcal{V}|}{k} \rceil$, confirming that $\mathcal{V}''$ is cohesive. Moreover, $\boldsymbol{\alpha}(v_i) = 0$ for all $v_i \in \mathcal{V}''$ implies $W \cap \bigcup_{v_i \in \mathcal{V}''} \mathsf{A}_i = \emptyset$, contradicting JR.

Now, assume that W violates JR, witnessed by $\mathcal{V}' \subseteq \mathcal{V}$ and $c \in \bigcap_{v_i \in \mathcal{V}'} \mathsf{A}_i$ with $|\mathcal{V}'| \geq \lceil \frac{|\mathcal{V}|}{k} \rceil$ and $W \cap \mathsf{A}_i = \emptyset$ for all $v_i \in \mathcal{V}'$. Then, $\boldsymbol{\alpha}(v_i) = \mathsf{sc}_{\mathrm{CC}}(v_i, W) = 0$ for all $v_i \in \mathcal{V}'$, while for $W' = \{c\}$ we have $\mathsf{sc}_{\mathrm{CC}}(v_i, W') = 1 > \boldsymbol{\alpha}(v_i)$ for all $v_i \in \mathcal{V}'$. Moreover, W' is admissible for $\mathcal{V}'$ since $|\mathcal{V}'| \geq \lceil \frac{|\mathcal{V}|}{k} \rceil$ implies that $\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor \geq 1 = |W'|$. Hence, $\mathcal{V}'$ is $\mathsf{NTU}_{\mathrm{CC}}$-blocking $\boldsymbol{\alpha}$, witnessing that $\boldsymbol{\alpha} \notin \mathsf{NTU}_{\mathrm{CC}}$-core.

Therefore, $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{CC}}$-core if and only if W satisfies JR. □

Since every committee that satisfies JR induces a core-stable utility function, Proposition 5.20 immediately implies the following.

Algorithm 5.2: Compute a $\mathsf{NTU}_{\mathrm{CC}}$-core-stable utility function.

Input: A profile $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$ with $\mathcal{R} = (\mathsf{A}_i)_{v_i \in \mathcal{V}}$ and $k \in [|\mathcal{C}|]$.

$\mathcal{V}' \leftarrow \mathcal{V}$; $W \leftarrow \emptyset$;
Initialize $\boldsymbol{\alpha} \colon \mathcal{V} \to \mathbb{N}$ with $\boldsymbol{\alpha}(v_i) = 0$ for all $v_i \in \mathcal{V}$;
while $\mathcal{V}' \neq \emptyset$ **and** $|W| < k$ **do**
  Let $c \leftarrow \arg\max_{c_j \in \mathcal{C} \setminus W} |\mathsf{V}(c_j)|$; // Recall that $\mathsf{V}(c_j)$ denotes the set of voters that approve of c_j
  $W \leftarrow W \cup \{c\}$; $\mathcal{V}' \leftarrow \mathcal{V}' \setminus \mathsf{V}(c)$
  foreach $c_j \in \mathcal{C} \setminus W$ **do** $\mathsf{V}(c_j) \leftarrow \mathsf{V}(c_j) \setminus \mathsf{V}(c)$;
  foreach $v_i \in \mathsf{V}(c)$ **do** $\boldsymbol{\alpha}(v_i) \leftarrow 1$;
end
if $|W| < k$ **then** Pick $k - |W|$ arbitrary alternatives from $\mathcal{C} \setminus W$ and add them to W ;
return $(\boldsymbol{\alpha}, W)$

Corollary 5.21. *If W is a winning committee under a voting rule that satisfies JR, then the utility function $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}(v) = \mathsf{sc}_{CC}(v, W)$ lies in the NTU_{CC}-core. Therefore, every JR-satisfying voting rule yields a utility function in the NTU_{CC}-core.*

With the correspondence in Proposition 5.20, it suffices to compute a size-k committee W that satisfies JR, since the induced utility function then belongs to $\mathsf{NTU}_{\mathrm{CC}}$-core. It is known that CC, PAV, and a greedy variant of CC (also called GreedyAV [ABC$^+$17]) satisfy JR. However, both CC and PAV are computationally difficult while GreedyAV runs in polynomial time [ABC$^+$17]. For completeness, we present the greedy procedure in Algorithm 5.2 and give a self-contained proof of the JR property.

Theorem 5.22. *Algorithm 5.2 computes a utility function in NTU_{CC}-core in polynomial time, and hence NTU_{CC}-core is always non-empty.*

Proof. Clearly, Algorithm 5.2 runs in polynomial time. Let $(\boldsymbol{\alpha}, W)$ be the output. By construction, $|W| = k$ and $\boldsymbol{\alpha}(v_i) = 1$ if and only if $W \cap \mathsf{A}_i \neq \emptyset$. Thus W witnesses feasibility of $\boldsymbol{\alpha}$ for CC.

By Proposition 5.20, it suffices to show that W satisfies JR. Assume for contradiction that W violates JR. Then there exists a cohesive coalition $\mathcal{V}^\star \subseteq \mathcal{V}$ with $|\mathcal{V}^\star| \geq \lceil \frac{|\mathcal{V}|}{k} \rceil$ and some alternative $c^\star \in \bigcap_{v_i \in \mathcal{V}^\star} \mathsf{A}_i$ such that $W \cap \bigcup_{v_i \in \mathcal{V}^\star} \mathsf{A}_i = \emptyset$. In particular, no voter in $\mathcal{V}^\star$ is ever removed from $\mathcal{V}'$ during the algorithm, because a voter is removed only when she approves a chosen alternative.

Consider an arbitrary iteration $\ell \in \{1, \ldots, k\}$ of the **while**-loop (so $|W| < k$). At the beginning of iteration ℓ, all voters in $\mathcal{V}^\star$ are still in $\mathcal{V}'$, and all of them approve $c^\star$. Hence $c^\star$ is approved by at least $|\mathcal{V}^\star| \geq |\mathcal{V}|/k$ voters in the current $\mathcal{V}'$. Since the algorithm chooses an alternative c maximizing $|\mathsf{V}(c)|$ over $\mathcal{C} \setminus W$ and since $c^\star$ was never chosen, it

follows that the selected c in iteration ℓ is approved by at least $\lceil \frac{|\mathcal{V}|}{k} \rceil$ currently non-deleted voters, and these voters are removed from $\mathcal{V}'$ at the end of the iteration. This implies that at the end of the k-th iteration, all voters are removed, a contradiction to the assumption that $|\mathcal{V}^\star| \geq \lceil \frac{|\mathcal{V}|}{k} \rceil$ voters remain at the end of the algorithm. □

We illustrate how to execute Algorithm 5.2:

Example 5.23. *Consider the instance $(\mathcal{P}, k)$ from Example 5.6 with $k = 2$. Initially, alternative c_4 has the most approvers $|\mathsf{V}(c_4)| = 4$, so the algorithm selects c_4 (line 4). It then removes the covered voters $\mathsf{V}(c_4) = \{v_2, v_3, v_5, v_6\}$ from consideration and sets $\boldsymbol{\alpha}(v_i) = 1$ for each of them (line 5–7). Among the remaining voters $\{v_1, v_4\}$, alternatives c_1 and c_2 each have one approver, so the algorithm selects one of them—say c_1—completing the committee $W = \{c_1, c_4\}$. The resulting utility function is $\boldsymbol{\alpha} = (1, 1, 1, 0, 1, 1)$; analogously, choosing c_2 yields $(0, 1, 1, 1, 1, 1)$. In either case, exactly one voter receives utility 0, and the output lies in the NTU_{CC}-core by Theorem 5.22. Note that neither committee is CC-winning: The unique winning committee $W' = \{c_1, c_2\}$ achieves $\mathsf{sc}_{CC}(v, W') = 1$ for all $v \in \mathcal{V}$. Since a CC-winning committee satisfies JR, by Corollary 5.21, W' also induces a utility function in the core.*

5.4 Computational Issues

The preceding results settle existence for TU-AV, TU-SAV, and NTU-CC. A natural follow-up is computational: Even when stable outcomes exist, can they be found or verified efficiently? This question requires tools from computational complexity; see Section 2.2 for details.

As a reminder: DP captures 'NP-and-coNPtype' conditions (here: Check feasibility plus absence of a blocking witness); Θ_2^{P} captures problems solvable in polynomial time with only logarithmically many NP-oracle calls (here: We can binary-search the optimal grand-coalition score while checking feasibility of the induced inequality system); Δ_2^{P} captures problems solvable in polynomial time with polynomially many NP-oracle calls; and Σ_2^{P} corresponds to an $\exists\forall$-structure typical of 'there exists a feasible outcome such that no coalition has a blocking deviation.'

With these classes in hand, we can state precise complexity bounds for our core problems. At a high level, our upper bounds follow a common pattern: Non-membership in the core is certified by an explicit blocking coalition together with an admissible committee witnessing strict improvement (or, in the TU case, a violation of a coalition inequality). The more delicate containments (notably the Θ_2^{P} and Δ_2^{P} upper bounds for existence) rely on separating the task of computing the grand-coalition optimum score from the feasibility of the resulting system of core inequalities.

Theorem 5.24. *1. AV-TU-CV and SAV-TU-CV are in* coNP*, while PAV-TU-CV and CC-TU-CV are in* DP*.*

2. *For all* $\lambda \in \{AV, SAV, PAV, CC\}$, λ-NTU-*CV is in* DP.

3. *AV-*TU-*CE and CC-*TU-*CE are in* Θ_2^P, *while SAV-*TU-*CE and PAV-*TU-*CE are in* Δ_2^P. *Moreover, for all* $\lambda \in \{AV, SAV, PAV, CC\}$, λ-NTU-*CE is in* Σ_2^P.

Proof. Statement 1: A utility function $\boldsymbol{\alpha}$ is in the TU_λ-core if

(a) there exists $W \subseteq \mathcal{C}$ with $|W| = k$ such that $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = \mathsf{sc}_\lambda(\mathcal{V}, W)$, and

(b) there do not exist a coalition $\mathcal{V}' \subseteq \mathcal{V}$ and a witness committee W' of size $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ such that $\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) < \mathsf{sc}_\lambda(\mathcal{V}', W')$.

Note that condition (a) together with condition (b) checks whether (c) $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V})$: If $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) < \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V})$ then the grand coalition $\mathcal{V}$ blocks.

For $\lambda \in \{\mathrm{AV}, \mathrm{SAV}\}$, a pair $(\mathcal{V}', W')$ serves as a polynomial-size certificate for violating condition (b), and can be verified in polynomial time, and condition (c) can be checked in polynomial time. Hence the complement of λ-TU-CV is in NP, and therefore λ-TU-CV is in coNP.

For $\lambda \in \{\mathrm{CC}, \mathrm{PAV}\}$, λ-TU-CV is in DP. Indeed, verifying condition (a) is in NP, with a polynomial-size certificate given by a committee W witnessing $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = \mathsf{sc}_\lambda(\mathcal{V}, W)$. Verifying condition (b) is in coNP, since a polynomial-size certificate for its violation is a pair $(\mathcal{V}', W')$, where $\mathcal{V}'$ is TU_λ-blocking and W' witnesses it. By Papadimitriou [Pap94], a language P is in DP if there exist NP-languages Q_1, Q_2 such that $P = Q_1 \cap \overline{Q_2}$. Taking Q_1 to be condition (a) and Q_2 to be the complement of condition (b) yields λ-TU-CV$\in$ DP for $\lambda \in \{\mathrm{CC}, \mathrm{PAV}\}$.

Statement 2: A utility function $\boldsymbol{\alpha}$ is in the NTU_λ-core if

(a) there exists $W \subseteq \mathcal{C}$ with $|W| = k$ such that $\boldsymbol{\alpha}(v_i) = \mathsf{sc}_\lambda(v_i, W)$ for all $v_i \in \mathcal{V}$, and

(b) there do not exist a coalition $\mathcal{V}' \subseteq \mathcal{V}$ and a witness committee W' of size $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ such that $\boldsymbol{\alpha}(v_i) < \mathsf{sc}_\lambda(v_i, W')$ for all $v_i \in \mathcal{V}'$.

For all $\lambda \in \{\mathrm{AV}, \mathrm{SAV}, \mathrm{PAV}, \mathrm{CC}\}$, λ-NTU-CV is in DP: verifying condition (a) is in NP with certificate W, and violating condition (b) is in NP with certificate $(\mathcal{V}', W')$. The same DP characterization argument as in Statement 1 applies.

Statement 3: We first prove Θ_2^{P}-containment for λ-TU-CE for $\lambda \in \{\mathrm{AV}, \mathrm{CC}\}$ by adapting our ellipsoid method approach from Chapter 3 (see e.g., Theorem 3.21). The key idea is to

(i) first compute the score of a winning committee $s^\star = \mathsf{sc}_\lambda(\mathcal{V})$ using logarithmically many NP-oracle calls via binary search, and then

(ii) solve a *hinted* version of the CE-problem that additionally receives $s^\star$ as part of the input.

Step 1: computing $s^\star$ with logarithmically many NP-queries. Let M be a polynomially bounded upper bound on $s^\star$ (e.g., $M = |\mathcal{V}|k$ for AV and $M = |\mathcal{V}|$ for CC). For any threshold $t \in \{0, \dots, M\}$, the committee score problem

$$\exists W \subseteq \mathcal{C},\ |W| = k \colon \mathsf{sc}_\lambda(\mathcal{V}, W) \geq t.$$

is in NP (certificate: W), hence we can compute $s^\star$ by binary search using $O(\log M)$ NP-oracle calls.

Step 2: A hinted version of CE is in coNP via ellipsoid-based certificate. Define the hinted problem:

Hinted-λ-TU-CE
Input: An MW-instance $(\mathcal{P}, k)$ and an integer $s^\star$ such that $s^\star = \mathsf{sc}_\lambda(\mathcal{V})$.
Question: Is the TU_λ-core of $(\mathcal{P}, k)$ non-empty?

We express non-emptiness of the TU_λ-core as feasibility of a linear program (LP) with variables $\{\boldsymbol{\alpha}(v)\}_{v \in \mathcal{V}}$:

$$\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = s^\star, \tag{5.10}$$

$$\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) \geq \mathsf{sc}_\lambda(\mathcal{V}', W'), \qquad \text{for all } \mathcal{V}' \subseteq \mathcal{V} \text{ and all } W' \subseteq \mathcal{C} \text{ with } |W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor. \tag{5.11}$$

This LP has exponentially many constraints of type (5.11).

Claim 5.25. *Hinted-λ-TU-CE is in* coNP *for $\lambda \in \{AV, SAV, PAV, CC\}$.*

Proof. We show coNP-containment using the ellipsoid method; see Section 2.3.2 for more details. More specifically, we use the variant where the constraints are not given explicitly. Our proof proceeds in two parts: In part 1, we show that every no-instance of Hinted-λ-TU-CE admits a polynomial-size witness, and in part 2, we show how to verify the witness. The verification procedure utilizes the $s^\star$, the score of a winning committee, and therefore the coNP-membership does not necessarily generalize to the non-hinted version. The proof is analogous to Claim 3.22 on page 63, however, due to the underlying problem being a voting problem, we will still provide a detailed explanation.

Part 1. We start with an informal description: First, recall that there exists an LP formulation for every TU-game (cf. Section 2.4.2). Suppose we are given a separation oracle (cf. Definition 2.39) as a blackbox algorithm. Then, by Theorem 2.40, the ellipsoid method would solve the problem in polynomial time. In the case of a no-instance, the separation oracle calls always return violated constraints, which consist of a coalition

of voters $\mathcal{V}'$ and the score $\mathsf{sc}_\lambda(\mathcal{V}')$. Informally, our witness will consist of a sequence of coalitions together with committees that witness the coalitions to be TU_λ-blocking. These witnesses will replace our separation oracle calls, and thereby allow us to execute the ellipsoid method with all our separation oracle calls already being available in the witness.

Let $((\mathcal{P}, k), s^\star)$ be a no-instance of Hinted-λ-TU-CE. Observe that a utility function $\boldsymbol{\alpha}$ is core-stable if and only if it satisfies the core stability constraints from Definition 2.51[8]. In other words, $\boldsymbol{\alpha}$ is core-stable if and only if it belongs to the polytope

$$P \coloneqq \left\{ \boldsymbol{\alpha} \in \mathbb{R}^n_{\geq 0} \;\middle|\; \sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = s^\star \text{ and } \sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) \geq \mathsf{sc}_\lambda(\mathcal{V}') \; \forall \mathcal{V}' \subseteq \mathcal{V} \right\}. \tag{5.12}$$

Note that we describe $\boldsymbol{\alpha}$ as a vector, rather than as a function in the polytope description. This can easily be seen to be equivalent by assigning an index to each voter, and saying $\boldsymbol{\alpha}(v)$ corresponds to the entry at the index of the voter in the vector. Thus, $((\mathcal{P}, k), s^\star)$ is a yes-instance of Hinted-λ-TU-CE if and only if $P \neq \emptyset$. The polytope P is given implicitly through the instance $(\mathcal{P}, k)$, and therefore, we use the separation oracle version of the ellipsoid method.

Let Q be a separation oracle for P. Given a utility function $\boldsymbol{\alpha}$, oracle Q either certifies that $\boldsymbol{\alpha} \in P$, or returns a violated constraint of P. In other words,

- if $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) \neq s^\star$, then Q returns the violated efficiency constraint (also preimputation constraint) $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = s^\star$; and
- otherwise, if there is a coalition $\mathcal{V}' \subseteq \mathcal{V}$ with $\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) < \mathsf{sc}_\lambda(\mathcal{V}')$, then Q returns the violated coalition constraint $\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) \geq \mathsf{sc}_\lambda(\mathcal{V}')$.

Since $(\mathcal{P}, k)$ does not admit a core-stable utility function, every query made by the ellipsoid method yields a violated constraint. Let

$$\mathcal{T} \coloneqq ((\boldsymbol{\alpha}^1, \omega^1), \ldots, (\boldsymbol{\alpha}^r, \omega^r))$$

denote the transcript of one execution of the ellipsoid method on $((\mathcal{P}, k), s^\star)$, where $\boldsymbol{\alpha}^j$ is the j-th queried utility function and ω^j is the violated constraint returned by Q. By Theorem 2.40, the ellipsoid method queries the separation oracle a polynomial number of times, as both the dimension (the number of voters in our case) and the maximum encoding length of a constraint[9] are bounded by the input size. Hence, r is polynomially bounded in n, the number of voters.

The transcript $\mathcal{T}$ is not directly verifiable in polynomial time: Checking that a coalition constraint ω^j, i.e., $\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) \geq \mathsf{sc}_\lambda(\mathcal{V}')$, is violated at $\boldsymbol{\alpha}^j$ requires computing the

[8] Section 2.4 also provides an explicit LP formulation.

[9] Remember that the longest constraint is of the form $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = s^\star$.

coalition value, which is NP-hard to compute for PAV and CC. In each step, we therefore attach a witness of the violation: a TU_λ-blocking voter coalition $\mathcal{V}'_j$, and a committee W'_j witnessing $\mathcal{V}'_j$ being TU_λ-blocking.

As $s^\star$ is part of the input, a violated efficiency constraint is recognized directly by testing $\sum_{v\in\mathcal{V}} \boldsymbol{\alpha}^j(v) = s^\star$ and needs no witness; we henceforth assume that every ω^j is a coalition constraint. This yields the witness

$$\mathcal{W} \coloneqq ((\mathcal{V}'_1, W'_1), \ldots, (\mathcal{V}'_r, W'_r))$$

of size polynomial in n.

Part 2. Assume we are given $\mathcal{W}$ as a witness that $(\mathcal{P}, k)$ does not admit a core-stable utility function. We verify $\mathcal{W}$ by simulating the deterministic ellipsoid method on $(\mathcal{P}, k)$, reconstructing the transcript $\mathcal{T}$ step by step. Whenever the simulation queries a utility function $\boldsymbol{\alpha}^j$, we first use the input value $s^\star$ to test whether $\sum_{v\in\mathcal{V}} \boldsymbol{\alpha}^j(v) = s^\star$; if not, the violated constraint ω^j is the efficiency constraint $\sum_{v\in\mathcal{V}} \boldsymbol{\alpha}^j(v) = s^\star$. Otherwise, we take the entry $(\mathcal{V}'_j, W'_j)$ of $\mathcal{W}$ and verify in polynomial time that W'_j is a witness to $\mathcal{V}'_j$ being TU_λ-blocking. If it is, the coalition constraint $\omega^j = \sum_{v\in\mathcal{V}'_j} \boldsymbol{\alpha}^j(v) \geq \mathsf{sc}_\lambda(\mathcal{V}'_j)$ is violated by $\boldsymbol{\alpha}^j$; that is, $\mathcal{V}'_j$ blocks $\boldsymbol{\alpha}^j$. We supply ω^j to the simulated ellipsoid method. The verifier accepts if and only if the simulation concludes that P is empty. As each of the r steps takes polynomial time and r is polynomially bounded, the verifier runs in polynomial time. (end of the proof of Claim 5.25 $\diamond$)

Step 3: concluding Θ_2^{P}. After we have determined the optimal score $s^\star = \mathsf{sc}_\lambda(\mathcal{V})$ using logarithmically many NP-oracle calls, we make one additional NP-oracle call to the complement of Hinted-λ-TU-CE on input $(\mathcal{P}, k, s^\star)$, which is in NP by Claim 5.25. If that call answers "yes" (the hinted core is empty), we reject; otherwise, we accept. In total we use $O(\log M) + 1$ many NP-oracle calls, and therefore λ-TU-CE is in Θ_2^{P}.

Δ_2^{P}-containment of λ-TU-CE for $\lambda \in \{\mathrm{SAV}, \mathrm{PAV}\}$ follows in the same manner, once the scores are made integral. Let $D \coloneqq \mathrm{lcm}(1, \ldots, k)$ for PAV and $D \coloneqq \mathrm{lcm}\{|\mathsf{A}_i| \mid \mathsf{A}_i \neq \emptyset\}$ for SAV. Let $M \coloneqq |\mathcal{V}| \sum_{i=1}^{k} \frac{1}{i}$ for PAV and $M \coloneqq |\mathcal{V}|$ for SAV. Every score $\mathsf{sc}_\lambda(\mathcal{V}', W')$ is an integer multiple of $\frac{1}{D}$, so multiplying all scores by D yields an equivalent instance with integral scores: The characteristic function, and hence the core, is scaled by D, so non-emptiness is preserved. Binary search on $\{0, \ldots, DM\}$ therefore determines $Ds^\star$, and thus $s^\star$, exactly, using $O(\log(DM))$ NP-oracle calls. However, $\log D = \Theta(k)$ for PAV and $\log D = \Theta(m)$ for SAV in the worst case, so D takes polynomially many bits and this is polynomially—but not logarithmically—many calls. As the remaining steps apply in the same way to the scaled instance (with hint $Ds^\star$), Δ_2^{P}-containment follows.

It remains to show Σ_2^{P}-containment for λ-NTU-CE. We do so by giving a non-deterministic polynomial-time algorithm with access to an NP-oracle.

Let $I = (\mathcal{P}, k)$ be an instance. Notice that I is a yes-instance if and only if there exists a size-k committee W such that the induced utility function

$$\boldsymbol{\alpha}(v_i) \;=\; \mathsf{sc}_\lambda(v_i, W) \qquad \text{for all } v_i \in \mathcal{V}$$

is in the core, i.e., admits no blocking voter coalition.

Our algorithm proceeds as follows.

1. Non-deterministically guess a size-k committee W, and compute the induced utility function $\boldsymbol{\alpha}$ by setting $\boldsymbol{\alpha}(v_i) = \mathsf{sc}_\lambda(v_i, W)$ for all $v_i \in \mathcal{V}$.
2. Query an NP-oracle Q_{block} that decides the complement of CV on $(I, \boldsymbol{\alpha})$, i.e., whether $\boldsymbol{\alpha}$ is blocked by some coalition. Accept if and only if Q_{block} answers NO.

Oracle. The oracle Q_{block} is an NP-oracle: a yes-certificate consists of a coalition $\mathcal{V}' \subseteq \mathcal{V}$, and the additional witness required by the definition of blocking under λ. The verification that $\mathcal{V}'$ blocks the induced utility function can be carried out in polynomial time under λ.

Correctness. If I is a yes-instance, then there exists a size-k committee $W^\star$ whose induced utility function $\boldsymbol{\alpha}^\star$ is in the core. On the computation branch that guesses $W^\star$, there is no blocking coalition for $\boldsymbol{\alpha}^\star$, hence Q_{block} answers NO on input $(I, \boldsymbol{\alpha}^\star)$; thus the algorithm accepts.

Conversely, if I is a no-instance, then for every size-k committee W the induced utility function is not in the core, so there exists a blocking coalition. Hence Q_{block} answers YES on $(I, \boldsymbol{\alpha})$ for every guessed W, and the algorithm rejects.

Running time. Guessing W and computing $\boldsymbol{\alpha}$ take polynomial time, and we make a single call to the NP-oracle. Therefore λ-NTU-CE $\in \mathrm{NP}^{\mathrm{NP}} = \Sigma_2^{\mathrm{P}}$. □

We next show that most of these upper bounds are essentially tight. The reductions isolate the source of difficulty in our setting: Blocking combines proportional feasibility constraints with an optimization problem over committees, and the interaction can encode both existential and universal structure (e.g., "there exists a blocking coalition" versus "for all coalitions no blocking exists"). As a result, core membership problems often land in classes such as DP, and core non-emptiness problems can require Θ_2^{P}-, Δ_2^{P}-, or Σ_2^{P}-style access to an NP-oracle.

5.4.1 Complexity for the TU Model

We begin with the TU model. In this case, blocking reduces to a *scalar* comparison: A coalition blocks a candidate utility function if it can achieve a strictly larger *total* score than the total utility assigned to its members. This makes CE resemble checking a system of core inequalities, but the coalition values are themselves defined by an

optimization problem over admissible committees. Accordingly, the complexity depends strongly on the underlying rule: For AV and SAV, total separability yields substantially more tractability, while for CC and PAV the induced optimization structure leads to hardness because computing a winning committee under both rules is already NP-hard [AS99, LB11, PRZ08, AGG+15].

Theorem 5.26. *AV-*TU*-CE and SAV-*TU*-CE are in* P*, while CC-*TU*-CE and PAV-*TU*-CE are* NP*-hard.*

Proof. The statement for AV and SAV follows directly from Theorem 5.10, as the core is always non-empty for these rules. The NP-hardness for CC and PAV comes from the fact that computing a corresponding winning committee is already NP-hard. For the sake of completeness, we give the corresponding reduction, which works for both. We reduce from the NP-complete problem RX3C; see Section 2.7.

Let $I = (X, \mathcal{S})$ be an instance of RX3C. We create an instance $I' = (\mathcal{P}, k)$ of CC-TU-CE (resp. PAV-TU-CE). Note that the two instances are the same.

Profile construction. Let $k \coloneqq \hat{n}$.

- For each set $S_i \in \mathcal{S}$, create an alternative c_i.
- For each element $x_i \in X$, create a voter v_i. Their approvals are $\mathsf{A}(v_i) \coloneqq \{c_j \mid x_i \in S_j\}$.

Thus, the total number of voters is $|\mathcal{V}| = 3\hat{n}$.

Correctness. We show that I is a yes-instance of RX3C if and only if the core is non-empty (for both CC and PAV).

For the "if" direction, we prove the contrapositive that "if I is a no-instance, then the core is empty". Assume that I is a no-instance, i.e., there does not exist an exact cover. Then, no size-k committee can be approved by all voters, and hence $\mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) < 3\hat{n}$.

We next argue that $\mathsf{TU}_{\mathrm{PAV}}\text{-}\chi(\mathcal{V}) < 3\hat{n}$ as well. Fix a committee W with $|W| = k = \hat{n}$ and write $t_i \coloneqq |\mathsf{A}(v_i) \cap W|$ for each voter v_i. Since every alternative is approved by exactly three voters, the total number of approval incidences inside W is $\sum_{i=1}^{3\hat{n}} t_i = 3|W| = 3\hat{n}$. Because I is a no-instance, W cannot cover all voters; let $d \geq 1$ be the number of uncovered voters (i.e., voters with $t_i = 0$). Then exactly $3\hat{n} - d$ voters are covered, and assigning one approved winner to each covered voter uses $3\hat{n} - d$ incidences. Hence, there remain precisely d further incidences, each corresponding to an *additional* approved winner for a voter who already has at least one.

Under PAV, the marginal gain of giving a voter an additional approved winner beyond their first is at most $1/2$ (the second winner contributes $1/2$ and the third contributes

1/3). Therefore,

$$\mathsf{sc}_{\mathrm{PAV}}(\mathcal{V}, W) \;\leq\; (3\hat{n} - d) \cdot 1 \;+\; d \cdot \frac{1}{2} \;=\; 3\hat{n} - \frac{d}{2} \;<\; 3\hat{n}.$$

Since W was arbitrary, it follows that $\mathsf{TU}_{\mathrm{PAV}}\text{-}\chi(\mathcal{V}) < 3\hat{n}$.

We utilize the Bondareva–Shapley Theorem (cf. Proposition 5.7) and show that there exists a balanced vector that violates (5.5). For each 3-set $S_j \in \mathcal{S}$, let $\mathcal{V}_j \coloneqq \{v_i \in \mathcal{V} \mid x_i \in S_j\}$ be the corresponding size-3 coalition (equivalently, $\mathcal{V}_j$ is exactly the set of voters approving c_j). Define a weight vector $\boldsymbol{\lambda} = (\lambda_{\mathcal{V}'})_{\mathcal{V}' \subseteq \mathcal{V}}$ by

$$\lambda_{\mathcal{V}'} \coloneqq \begin{cases} \frac{1}{3}, & \text{if } \mathcal{V}' = \mathcal{V}_j \text{ for some } j \in [\hat{m}], \\ 0, & \text{otherwise.} \end{cases}$$

This vector is balanced: Each voter v_i (corresponding to element x_i) belongs to exactly three coalitions $\mathcal{V}_j$ because x_i appears in exactly three 3-sets of $\mathcal{S}$, and each such coalition has weight 1/3. Therefore $\sum_{\mathcal{V}' \ni v_i} \lambda_{\mathcal{V}'} = 3 \cdot \frac{1}{3} = 1$, i.e., (5.4) holds.

Moreover, every coalition $\mathcal{V}_j$ with non-zero weight has size 3 and can select exactly one alternative (since $\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor = \lfloor 3/3 \rfloor = 1$). Choosing c_j, all three members approve it, hence

$$\mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}_j) = 3 \qquad \text{and} \qquad \mathsf{TU}_{\mathrm{PAV}}\text{-}\chi(\mathcal{V}_j) = 3.$$

Thus, for $\lambda \in \{\mathrm{CC}, \mathrm{PAV}\}$ we obtain

$$\sum_{\mathcal{V}' \subseteq \mathcal{V}} \lambda_{\mathcal{V}'} \cdot \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V}') \;=\; \sum_{j=1}^{\hat{m}} \frac{1}{3} \cdot \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V}_j) \;=\; \sum_{j=1}^{\hat{m}} \frac{1}{3} \cdot 3 \;=\; \hat{m} \;=\; 3\hat{n}.$$

Combining this with $\mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) < 3\hat{n}$ and $\mathsf{TU}_{\mathrm{PAV}}\text{-}\chi(\mathcal{V}) < 3\hat{n}$ shown above, we get

$$\sum_{\mathcal{V}' \subseteq \mathcal{V}} \lambda_{\mathcal{V}'} \cdot \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V}') \;>\; \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V}) \qquad \text{for } \lambda \in \{\mathrm{CC}, \mathrm{PAV}\},$$

so (5.5) is violated. By the Bondareva–Shapley Theorem (Proposition 5.7), the core is empty. This completes the contrapositive and hence the "if" direction.

For the "only if" direction, assume that I is a yes-instance and let $\mathcal{S}'$ be an exact cover. Define $W \coloneqq \{c_j \mid S_j \in \mathcal{S}'\}$, so $|W| = \hat{n}$ and every voter approves exactly one alternative in W. Hence $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}, W) = |\mathcal{V}|$ and $\mathsf{sc}_{\mathrm{PAV}}(\mathcal{V}, W) = |\mathcal{V}|$. Since $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}, W) \leq |\mathcal{V}|$ for every committee and (by the argument in the "if" direction for PAV) any size-$\hat{n}$ committee that leaves some voter uncovered has PAV-score strictly less than $|\mathcal{V}|$, it follows that

$$\mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) = |\mathcal{V}| \qquad \text{and} \qquad \mathsf{TU}_{\mathrm{PAV}}\text{-}\chi(\mathcal{V}) = |\mathcal{V}|.$$

Consider the utility function $\boldsymbol{\alpha}(v) = 1$ for all $v \in \mathcal{V}$. Then $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = \mathsf{TU}_\lambda\text{-}\chi(\mathcal{V})$ for $\lambda \in \{\mathrm{CC}, \mathrm{PAV}\}$.

It remains to show that no coalition $\mathcal{V}' \subseteq \mathcal{V}$ blocks $\boldsymbol{\alpha}$. Let $\mathcal{V}' \subseteq \mathcal{V}$ be arbitrary and let $\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor := \left\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \right\rfloor = \left\lfloor \frac{|\mathcal{V}'|}{3} \right\rfloor$. For any witness committee W' with $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$, we have

$$\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}', W') \leq 3|W'|,$$

because each alternative is approved by exactly three voters. Moreover,

$$\mathsf{sc}_{\mathrm{PAV}}(\mathcal{V}', W') \leq 3|W'|,$$

because adding an alternative can increase each approving voter's PAV-utility by at most 1, and each alternative is approved by exactly three voters. Therefore, for $\lambda \in \{\mathrm{CC}, \mathrm{PAV}\}$,

$$\mathsf{sc}_{\lambda}(\mathcal{V}', W') \leq 3|W'| \leq 3\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor \leq |\mathcal{V}'| = \sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v).$$

Taking the maximum over all feasible W' yields $\mathsf{TU}_{\lambda}\text{-}\chi(\mathcal{V}') \leq \sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v)$, so $\mathcal{V}'$ cannot block. Thus $\boldsymbol{\alpha} \in \mathsf{TU}_{\mathrm{CC}}$-core and $\boldsymbol{\alpha} \in \mathsf{TU}_{\mathrm{PAV}}$-core. This concludes the proof. □

We next turn from CE to CV. In the TU setting, a blocking certificate consists of a coalition together with an admissible committee that achieves a larger total score, turning non-membership into a succinct witness. This perspective yields the stated upper bounds and guides the hardness reductions below.

The hardness for AV, and later for SAV via a slight modification, is via reducing from the complement of the NP-complete problem Balanced Complete Bipartite Subgraph (Biclique); see Section 2.7. As a reminder: Given a graph $G = (U, E)$ and a non-negative integer h, Biclique asks whether there exist two disjoint subsets $U_1, U_2 \subseteq U$ with $U_1 \cap U_2 = \emptyset$ and $|U_1| = |U_2| = h$ such that $\{u, w\} \in E$ for all $u \in U_1$ and all $w \in U_2$[10].

Theorem 5.27. *AV-TU-CV is* coNP*-complete.*

Proof. CoNP-membership follows from Theorem 5.24.1. To show coNP-hardness, we reduce from Biclique. Let $I = (G = (U, E), h)$ be an instance with $|U| = \hat{n}$ and assume without loss of generality that $\hat{n} \geq 2h$ and $\hat{n} \geq 2$. We construct an instance $I' = (\mathcal{P}, k, \boldsymbol{\alpha})$ of AV-TU-CV.

Profile construction. Let $L := \hat{n}^2 + 1$ and $k := L + h$.

- For each vertex $u_i \in U$, create a *vertex-alternative* c_i; let $\mathcal{C}_U := \{c_i \mid u_i \in U\}$.
- Create $k = L + h$ *dummy alternatives* $\mathcal{C}_D := \{d_1, \ldots, d_{L+h}\}$.
- For each vertex $u_i \in U$, create a *vertex-voter* v_i; let $\mathcal{V}_U := \{v_i \mid u_i \in U\}$. Their approvals are $\mathsf{A}(v_i) := \{c_j \mid \{u_i, u_j\} \in E\}$; in particular, $c_i \notin \mathsf{A}(v_i)$ since G is simple.

[10] We use U for the vertex set in order to not introduce confusion with the set of voters $\mathcal{V}$.

- Create $d_1 \coloneqq L \cdot \hat{n}^2 - (\hat{n} - h)$ *full voters* $\mathcal{V}_O \coloneqq \{o_1, \dots, o_{d_1}\}$. Each full voter o_i approves all dummy alternatives: $\mathsf{A}(o_i) \coloneqq \mathcal{C}_D$.
- Create $d_2 \coloneqq h \cdot \hat{n}^2 - h$ *empty voters* $\mathcal{V}_B \coloneqq \{b_1, \dots, b_{d_2}\}$. Each empty voter b_i approves nothing: $\mathsf{A}(b_i) \coloneqq \emptyset$.

The total number of voters is $|\mathcal{V}| = |\mathcal{V}_U| + |\mathcal{V}_O| + |\mathcal{V}_B| = \hat{n} + (L\hat{n}^2 - (\hat{n} - h)) + (h\hat{n}^2 - h) = k \cdot \hat{n}^2$.

The utility function $\boldsymbol{\alpha}$. Define $\varepsilon \coloneqq \frac{\hat{n}h - 1}{d_1}$ (note that $\varepsilon < 1$ since $\hat{n}^2 - 1 > \hat{n} - h$ and hence $d_1 > L = \hat{n}^2 + 1$). Set $\boldsymbol{\alpha}(v_i) = h - \frac{1}{\hat{n}}$, $\forall v_i \in \mathcal{V}_U$, $\boldsymbol{\alpha}(o_i) = L + h - \varepsilon$, $\forall o_i \in \mathcal{V}_O$, and $\boldsymbol{\alpha}(b_i) = 0$, $\forall b_i \in \mathcal{V}_B$.

This completes the construction of the instance. Before we show the correctness, we first observe that the utility function $\boldsymbol{\alpha}$ is feasible and optimal for $\mathcal{V}$.

Claim 5.28. $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = \mathit{TU}_{\mathit{AV}}\text{-}\chi(\mathcal{V})$.

Proof. Each dummy alternative $d \in \mathcal{C}_D$ is approved by all d_1 full voters and by no other voters, hence $|\mathsf{V}(d)| = d_1 = L\hat{n}^2 - (\hat{n} - h)$. Each vertex-alternative $c_j \in \mathcal{C}_U$ is approved only by some subset of the $\hat{n}$ vertex voters, hence $|\mathsf{V}(c_j)| \leq \hat{n} < d_1$. Therefore a winning committee under AV is $W^* = \mathcal{C}_D$ and

$$\mathsf{TU}_{\mathrm{AV}}\text{-}\chi(\mathcal{V}) = k \cdot d_1 = (L + h) \cdot (L\hat{n}^2 - (\hat{n} - h)).$$

On the other hand,

$$\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = \sum_{v_i \in \mathcal{V}_U} \left(h - \frac{1}{\hat{n}}\right) + \sum_{o_i \in \mathcal{V}_O} (L + h - \varepsilon) = (h\hat{n} - 1) + d_1(L + h) - d_1\varepsilon = d_1(L + h),$$

because $d_1\varepsilon = h\hat{n} - 1$ by definition of ε. (end of the proof of Claim 5.28 $\diamond$)

Correctness. By Claim 5.28, $\boldsymbol{\alpha}$ is feasible for $\mathcal{V}$, so $\boldsymbol{\alpha} \notin \mathsf{TU}_{\mathrm{AV}}$-core if and only if there exists a blocking coalition.

For the "if" direction (biclique implies blocking coalition), assume that I is a yes-instance and let U_1 and U_2 witness a balanced biclique: $U_1 \cap U_2 = \emptyset$, $|U_1| = |U_2| = h$, and all edges between U_1 and U_2 are present. Let $\mathcal{V}' \coloneqq \{v_i \mid u_i \in U_1\} \cup \mathcal{V}_B$. Then, $|\mathcal{V}'| = h + (h\hat{n}^2 - h) = h\hat{n}^2$, hence the coalition share is $\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor = \lfloor \frac{h\hat{n}^2 \cdot k}{k\hat{n}^2} \rfloor = h$. Let $W' \coloneqq \{c_j \mid u_j \in U_2\}$, so $|W'| = h$. Every voter in $\{v_i \mid u_i \in U_1\}$ approves every alternative in W', and hence $\mathsf{sc}_{\mathrm{AV}}(\mathcal{V}', W') = h \cdot h = h^2$. Since empty voters have zero utility, $\sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i) = \sum_{u_i \in U_1} (h - 1/\hat{n}) = h^2 - h/\hat{n} < h^2$. Thus, $\mathcal{V}'$ blocks $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha} \notin \mathsf{TU}_{\mathrm{AV}}$-core, i.e., I' is a no-instance.

For the "only if" direction (blocking coalition implies biclique), assume that I' is a no-instance and let $\mathcal{V}'$ be a blocking coalition and W' a witness: $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ and $\mathsf{sc}_{\mathrm{AV}}(\mathcal{V}', W') > \sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v)$.

Claim 5.29. *No blocking coalition contains any full voter, i.e.,* $\mathcal{V}' \cap \mathcal{V}_O = \emptyset$.

Proof. Towards a contradiction, suppose $x \coloneqq |\mathcal{V}' \cap \mathcal{V}_O| \geq 1$. Each dummy alternative has score exactly x in $\mathcal{V}_O$, while every vertex-alternative has score at most $|\mathcal{V}' \cap \mathcal{V}_U| \leq \hat{n}$. Let $t \coloneqq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor = \lfloor \frac{|\mathcal{V}'|}{\hat{n}^2} \rfloor$.

If $x > \hat{n}$, then any AV-score-maximizing witness W' committee may be assumed to contain only dummy alternatives (as otherwise replacing one vertex-alternative with a dummy alternative would increase the score), so $\mathsf{sc}_{\mathrm{AV}}(\mathcal{V}', W') \leq tx$.

Blocking would imply $tx > x(L + h - \varepsilon)$ and hence $t > L + h - \varepsilon$. Since t is an integer, this yields $t \geq L + h = k$ and therefore $|\mathcal{V}'| \geq \hat{n}^2 k = |\mathcal{V}|$, so $\mathcal{V}' = \mathcal{V}$. But the grand coalition $\mathcal{V}$ does not block $\boldsymbol{\alpha}$ (Claim 5.28).

If $1 \leq x \leq \hat{n}$, then the total AV-score of any committee of size at most t is at most $t \cdot \hat{n}$, because every alternative has AV-score at most $\hat{n}$ in $\mathcal{V}'$. Yet one full voter contributes utility $\boldsymbol{\alpha}(o_i) = L + h - \varepsilon > \hat{n}^2 \geq t \cdot \hat{n}$ (since $|\mathcal{V}'| \leq h \cdot \hat{n}^2 + 2\hat{n} - h$, due to $|\mathcal{V}' \cap \mathcal{V}_O| \leq \hat{n}$, and therefore $t \leq h + 1 < \hat{n}$). Hence $\mathcal{V}'$ cannot block $\boldsymbol{\alpha}$.

In both cases we obtain a contradiction, so $\mathcal{V}' \cap \mathcal{V}_O = \emptyset$. (end of the proof of Claim 5.29 $\diamond$)

Claim 5.30. *If* $\mathcal{V}'$ *blocks* $\boldsymbol{\alpha}$*, then* $\mathcal{V}' \cup \mathcal{V}_B$ *also blocks* $\boldsymbol{\alpha}$ *(with the same witness committee).*

Proof. Empty voters approve no alternatives and have utility 0, hence adding them does not change the AV-score of W' nor the utility sum $\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v)$, while it can only increase the share $\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$. (end of the proof of Claim 5.30 $\diamond$)

By Claims 5.29 and 5.30, without loss of generality, we may assume $\mathcal{V}_B \subseteq \mathcal{V}'$ and $\mathcal{V}' \cap \mathcal{V}_O = \emptyset$. Let $h' \coloneqq |\mathcal{V}' \cap \mathcal{V}_U|$ be the number of vertex-voters in $\mathcal{V}'$. We aim to show that $h' \geq h$ and that the vertex-voters in the blocking coalition and vertex-alternatives in the witness correspond to a balanced biclique of size at least h on each side. First, $|\mathcal{V}'| = |\mathcal{V}_B| + h' = (h\hat{n}^2 - h) + h'$. Since $0 \leq h' \leq \hat{n} < \hat{n}^2$, it follows that if $h' < h$, then $\lfloor \frac{|\mathcal{V}'|}{\hat{n}^2} \rfloor = h - 1$; otherwise $\lfloor \frac{|\mathcal{V}'|}{\hat{n}^2} \rfloor = h$.

If $h' < h$, then $|W'| \leq h - 1$ and every alternative has AV-score at most h' in $\mathcal{V}'$, hence $\mathsf{sc}_{\mathrm{AV}}(\mathcal{V}', W') \leq (h-1)h'$. But the coalition utility is $\sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i) = h' \left(h - \frac{1}{\hat{n}}\right) > (h-1)h'$, a contradiction. Therefore $h' \geq h$ and the budget forces $|W'| \leq h$.

Now note that for every size-at-most-h committee W'' we have $\mathsf{sc}_{\mathrm{AV}}(\mathcal{V}', W'') \leq hh'$ since each chosen alternative has score at most h'. Since $\mathcal{V}'$ is blocking, we get

$$\mathsf{sc}_{\mathrm{AV}}(\mathcal{V}', W') > h'(h - \tfrac{1}{\hat{n}}) = h'h - \tfrac{h'}{\hat{n}}.$$

The left-hand side is an integer and the right-hand side is strictly less than $h'h$, so $\mathsf{sc}_{\mathrm{AV}}(\mathcal{V}', W') \geq h'h$. Thus $\mathsf{sc}_{\mathrm{AV}}(\mathcal{V}', W') = h'h$ and $|W'| = h$, and consequently every

alternative in W' has score exactly h' in $\mathcal{V}'$, i.e. it is approved by *all* h' vertex voters in $\mathcal{V}'$.

Let U_1 be any subset of size-h of the vertices corresponding to $\mathcal{V}' \cap \mathcal{V}_U$, and let U_2 be the set of size-h of vertices corresponding to W'. Then every vertex in U_1 is adjacent to every vertex in U_2. Moreover $U_1 \cap U_2 = \emptyset$ because a voter v_i does not approve c_i (no self-loop), so if $c_i \in W'$ and $v_i \in \mathcal{V}'$, then c_i would not be approved by all voters in $\mathcal{V}' \cap \mathcal{V}_U$, contradicting that c_i has score h'. Hence (U_1, U_2) witnesses that I is a yes-instance of Biclique.

We have shown that I is a yes-instance if and only if I' is a no-instance. Therefore AV-TU-CV is coNP-hard. Together with coNP-containment in Theorem 5.24.1, the problem is coNP-complete. □

The following result can be obtained using a similar reduction to Theorem 5.27. Intuitively, we circumvent the fact that the utility that a voter v obtains from an approved alternative is $\frac{1}{|\mathsf{A}(v)|}$ rather than 1 by making sure that all voters that approve at least one alternative approve the exact same number of alternatives by adding dummy alternatives that are approved by one voter each.

Theorem 5.31. *SAV-TU-CV is coNP-complete.*

Proof. CoNP-membership follows from Theorem 5.24.1. To show coNP-hardness, we reduce from Biclique. Let $I = (G = (U, E), h)$ be an instance with $|U| = \hat{n}$ and assume without loss of generality that $\hat{n} \geq 2h$ and $\hat{n} \geq 2$. We construct an instance $I' = (\mathcal{P}, k, \boldsymbol{\alpha})$ of SAV-TU-CV.

Profile construction. Let $L \coloneqq \hat{n}^2 + 1$ and $k \coloneqq L + h$.

- For each vertex $u_i \in U$, create a *vertex-alternative* c_i; let $\mathcal{C}_U \coloneqq \{c_i \mid u_i \in U\}$.
- For each vertex $u_i \in U$, create $k - \deg(u_i) = L + h - \deg(u_i)$ *private alternatives* $p^i_1, \ldots, p^i_{k-\deg(u_i)}$; let $P \coloneqq \{p^i_j \mid u_i \in U, j \in [k - \deg(u_i)]\}$.
- Create $k = L + h$ *dummy alternatives* $\mathcal{C}_D \coloneqq \{d_1, \ldots, d_{L+h}\}$.
- For each vertex $u_i \in U$, create a *vertex-voter* v_i; let $\mathcal{V}_U \coloneqq \{v_i \mid u_i \in U\}$. Their approvals are $\mathsf{A}(v_i) \coloneqq \{c_j \mid \{u_i, u_j\} \in E\} \cup \{p^i_1, \ldots, p^i_{k-\deg(u_i)}\}$; in particular, $c_i \notin \mathsf{A}(v_i)$ since G is simple.
- Create $d_1 \coloneqq L \cdot \hat{n}^2 - (\hat{n} - h)$ *full voters* $\mathcal{V}_O \coloneqq \{o_1, \ldots, o_{d_1}\}$. Each full voter o_i approves all dummy alternatives: $\mathsf{A}(o_i) \coloneqq \mathcal{C}_D$.
- Create $d_2 \coloneqq h \cdot \hat{n}^2 - h$ *empty voters* $\mathcal{V}_B \coloneqq \{b_1, \ldots, b_{d_2}\}$. Each empty voter b_i approves nothing: $\mathsf{A}(b_i) \coloneqq \emptyset$.

The total number of voters is $|\mathcal{V}| = |\mathcal{V}_U| + |\mathcal{V}_O| + |\mathcal{V}_B| = \hat{n} + (L\hat{n}^2 - (\hat{n} - h)) + (h\hat{n}^2 - h) = k \cdot \hat{n}^2$.

The utility function $\boldsymbol{\alpha}$. Define $\varepsilon \coloneqq \frac{\hat{n}h-1}{d_1}$ (note that $\varepsilon < 1$ since $\hat{n}^2 - 1 > \hat{n} - h$ and hence $d_1 > L = \hat{n}^2 + 1$). Set $\boldsymbol{\alpha}(v_i) = \frac{h-\frac{1}{\hat{n}}}{k}$, $\forall v_i \in \mathcal{V}_U$, $\boldsymbol{\alpha}(o_i) = \frac{L+h-\varepsilon}{k}$, $\forall o_i \in \mathcal{V}_O$, and $\boldsymbol{\alpha}(b_i) = 0$, $\forall b_i \in \mathcal{V}_B$.

This completes the construction of the instance. An attentive reader may have noticed that the construction of the profile and utility function is very similar to the one in Theorem 5.27. The only differences lie in the fact that we added the private alternatives and divided the utility of each voter by k. Due to this, the remainder of the proof will follow the same steps as the proof of Theorem 5.27.

We can immediately observe the following from the construction:

Observation 5.32. *Each voter approves either* 0 *or* k *alternatives. Therefore it holds that* $\mathsf{TU}_{SAV}\text{-}\chi(\mathcal{V}') = \frac{\mathsf{TU}_{AV}\text{-}\chi(\mathcal{V}')}{k}$ *and* $\mathsf{sc}_{SAV}(\mathcal{V}', W') = \frac{\mathsf{sc}_{AV}(\mathcal{V}', W')}{k}$ *for all* $\mathcal{V}' \subseteq \mathcal{V}$ *and* $W' \subseteq \mathcal{C}$.

Next, we observe that the utility function $\boldsymbol{\alpha}$ is feasible and optimal for $\mathcal{V}$.

Claim 5.33. $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = \mathsf{TU}_{SAV}\text{-}\chi(\mathcal{V})$.

Proof. Each dummy alternative $d \in \mathcal{C}_D$ is approved by all d_1 full voters and by no other voters, hence $|\mathsf{V}(d)| = d_1 = L\hat{n}^2 - (\hat{n} - h)$. Each vertex-alternative $c_j \in \mathcal{C}_U$ is approved only by some subset of the $\hat{n}$ vertex voters, hence $|\mathsf{V}(c_j)| \leq \hat{n} < d_1$. Each private alternative p_j^i is approved only by one voter, hence $|\mathsf{V}(p_j^i)| = 1$. Therefore a winning committee under SAV is $W^* = \mathcal{C}_D$ and by Observation 5.32

$$\mathsf{TU}_{\text{SAV}}\text{-}\chi(\mathcal{V}) = \frac{\mathsf{TU}_{\text{AV}}\text{-}\chi(\mathcal{V})}{k} = \frac{k \cdot d_1}{k} = d_1 = (L\hat{n}^2 - (\hat{n} - h)).$$

On the other hand,

$$\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = \sum_{v_i \in \mathcal{V}_U} \left(\frac{h - \frac{1}{\hat{n}}}{k} \right) + \sum_{o_i \in \mathcal{V}_O} \frac{(L + h - \varepsilon)}{k} = \frac{(h\hat{n} - 1) + d_1(L + h) - d_1\varepsilon}{k} = d_1,$$

because $d_1\varepsilon = h\hat{n} - 1$ by definition of ε. (end of the proof of Claim 5.33 $\diamond$)

Correctness. By Claim 5.33, $\boldsymbol{\alpha}$ is feasible for $\mathcal{V}$, so $\boldsymbol{\alpha} \notin \mathsf{TU}_{\text{SAV}}$-core if and only if there exists a blocking coalition.

For the "if" direction (biclique implies blocking coalition), assume that I is a yes-instance and let U_1 and U_2 witness a balanced biclique: $U_1 \cap U_2 = \emptyset$, $|U_1| = |U_2| = h$, and all edges between U_1 and U_2 are present. Let $\mathcal{V}' \coloneqq \{v_i \mid u_i \in U_1\} \cup \mathcal{V}_B$. Then, $|\mathcal{V}'| = h + (h\hat{n}^2 - h) = h\hat{n}^2$, hence the coalition share is $\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor = \lfloor \frac{h\hat{n}^2 \cdot k}{k\hat{n}^2} \rfloor = h$. Let $W' \coloneqq \{c_j \mid u_j \in U_2\}$, so $|W'| = h$. Every voter in $\{v_i \mid u_i \in U_1\}$ approves every

alternative in W', and hence $\mathsf{sc}_{\mathrm{SAV}}(\mathcal{V}', W') = \frac{\mathsf{sc}_{\mathrm{AV}}(\mathcal{V}', W')}{k} = \frac{h^2}{k}$. Since empty voters have zero utility, $\sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i) = \sum_{u_i \in U_1} \frac{h - 1/\hat{n}}{k} = \frac{h^2 - h/\hat{n}}{k} < \frac{h^2}{k}$. Thus, $\mathcal{V}'$ blocks $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha} \notin \mathsf{TU}_{\mathrm{SAV}}$-core, i.e., I' is a no-instance.

For the "only if" direction (blocking coalition implies biclique), assume that I' is a no-instance and let $\mathcal{V}'$ be a blocking coalition and W' a witness: $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ and $\mathsf{sc}_{\mathrm{SAV}}(\mathcal{V}', W') > \sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v)$.

Claim 5.34. *No blocking coalition contains any full voter, i.e., $\mathcal{V}' \cap \mathcal{V}_O = \emptyset$.*

Proof. Towards a contradiction, suppose $x \coloneqq |\mathcal{V}' \cap \mathcal{V}_O| \geq 1$. Each dummy alternative has score exactly $\frac{x}{k}$ in $\mathcal{V}_O$, while every vertex-alternative has score at most $\frac{|\mathcal{V}' \cap \mathcal{V}_U|}{k} \leq \frac{\hat{n}}{k}$ and every private alternative has score at most $\frac{1}{k}$. Let $t \coloneqq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor = \lfloor \frac{|\mathcal{V}'|}{\hat{n}^2} \rfloor$.

If $x > \hat{n}$, then any SAV-score-maximizing witness W' committee may be assumed to contain only dummy alternatives (as otherwise replacing one vertex-alternative or private alternative with a dummy alternative would increase the score), so $\mathsf{sc}_{\mathrm{SAV}}(\mathcal{V}', W') \leq \frac{tx}{k}$.

Blocking would imply $\frac{tx}{k} > \frac{x(L+h-\varepsilon)}{k}$ and hence $t > L + h - \varepsilon$. Since t is an integer, this yields $t \geq L + h = k$ and therefore $|\mathcal{V}'| \geq \hat{n}^2 k = |\mathcal{V}|$, so $\mathcal{V}' = \mathcal{V}$. But the grand coalition $\mathcal{V}$ does not block $\boldsymbol{\alpha}$ (Claim 5.33).

If $1 \leq x \leq \hat{n}$, then the total SAV-score of any committee of size at most t is at most $\frac{t \cdot \hat{n}}{k}$, because every alternative has SAV-score at most $\frac{\hat{n}}{k}$ in $\mathcal{V}'$. Yet one full voter contributes utility $\boldsymbol{\alpha}(o_i) = \frac{L+h-\varepsilon}{k} > \frac{\hat{n}^2}{k} \geq \frac{t \cdot \hat{n}}{k}$ (since $|\mathcal{V}'| \leq h \cdot \hat{n}^2 + 2\hat{n} - h$, due to $|\mathcal{V}' \cap \mathcal{V}_O| \leq \hat{n}$, and therefore $t \leq h + 1 < \hat{n}$). Hence $\mathcal{V}'$ cannot block $\boldsymbol{\alpha}$.

In both cases, we obtain a contradiction, so $\mathcal{V}' \cap \mathcal{V}_O = \emptyset$. (end of the proof of Claim 5.34 $\diamond$)

Claim 5.35. *If $\mathcal{V}'$ blocks $\boldsymbol{\alpha}$, then $\mathcal{V}' \cup \mathcal{V}_B$ also blocks $\boldsymbol{\alpha}$ (with the same witness committee).*

Proof. Empty voters approve no alternatives and have utility 0, hence adding them does not change the SAV-score nor the utility sum $\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v)$, while it can only increase the share $\lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$. (end of the proof of Claim 5.35 $\diamond$)

By Claims 5.34 and 5.35, without loss of generality, we may assume $\mathcal{V}_B \subseteq \mathcal{V}'$ and $\mathcal{V}' \cap \mathcal{V}_O = \emptyset$. Let $h' \coloneqq |\mathcal{V}' \cap \mathcal{V}_U|$ be the number of vertex-voters in $\mathcal{V}'$. We aim to show that $h' \geq h$ and that the vertex-voters in the blocking coalition and vertex-alternatives in the witness correspond to a balanced biclique of size at least h on each side. First, $|\mathcal{V}'| = |\mathcal{V}_B| + h' = (h\hat{n}^2 - h) + h'$. Since $0 \leq h' \leq \hat{n} < \hat{n}^2$, it follows that if $h' < h$, then $\lfloor \frac{|\mathcal{V}'|}{\hat{n}^2} \rfloor = h - 1$; otherwise $\lfloor \frac{|\mathcal{V}'|}{\hat{n}^2} \rfloor = h$.

If $h' < h$, then $|W'| \leq h - 1$ and every alternative has SAV-score at most $\frac{h'}{k}$ in $\mathcal{V}'$, hence $\mathsf{sc}_{\mathrm{SAV}}(\mathcal{V}', W') \leq \frac{(h-1)h'}{k}$. But the coalition utility is $\sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i) = \frac{h'\left(h - \frac{1}{\hat{n}}\right)}{k} > \frac{(h-1)h'}{k}$, a contradiction. Therefore $h' \geq h$ and the budget forces $|W'| \leq h$.

Now note that for every size-at-most-h committee W'' we have $\mathsf{sc}_{\mathrm{SAV}}(\mathcal{V}', W'') \leq \frac{hh'}{k}$ since each chosen alternative has score at most $\frac{h'}{k}$. Since $\mathcal{V}'$ is blocking, we get

$$\mathsf{sc}_{\mathrm{SAV}}(\mathcal{V}', W') > \frac{h'(h-\frac{1}{n})}{k} = \frac{h'h-\frac{h'}{n}}{k}.$$

If we multiply everything with k, then the left-hand side is an integer and the right-hand side is strictly less than $h'h$, so $\mathsf{sc}_{\mathrm{SAV}}(\mathcal{V}', W') \geq \frac{h'h}{k}$. Thus $\mathsf{sc}_{\mathrm{SAV}}(\mathcal{V}', W') = \frac{h'h}{k}$ and $|W'| = h$, and consequently every alternative in W' has score exactly h' in $\mathcal{V}'$, i.e., it is approved by *all* h' vertex voters in $\mathcal{V}'$.

Let U_1 be any subset of size-h of the vertices corresponding to $\mathcal{V}' \cap \mathcal{V}_U$, and let U_2 be the set of size-h of vertices corresponding to W'. Then every vertex in U_1 is adjacent to every vertex in U_2. Moreover $U_1 \cap U_2 = \emptyset$ because a voter v_i does not approve c_i (no self-loop), so if $c_i \in W'$ and $v_i \in \mathcal{V}'$, then c_i would not be approved by all voters in $\mathcal{V}' \cap \mathcal{V}_U$, contradicting that c_i has score h'. Hence (U_1, U_2) witnesses that I is a yes-instance of Biclique.

We have shown that I is a yes-instance if and only if I' is a no-instance. Therefore SAV-TU-CV is coNP-hard. Together with coNP-containment in Theorem 5.24.1, the problem is coNP-complete. □

We next turn to CC in the TU model. Unlike AV and SAV, core membership verification for CC combines

1. an NP-type witness of a winning committee and

2. a coNP-type requirement that no blocking coalition exists;

these two constraints naturally place the problem in the class DP. We show that this upper bound is tight by a reduction from a DP-hard problem.

Theorem 5.36. *CC-TU-CV is* DP*-complete.*

Proof. DP-membership follows from Theorem 5.24.1. Therefore, it suffices to show DP-hard-hardness. We reduce from RX3C-UNSAT.[11]

[11]This problem is DP-complete. Membership is immediate: RX3C is in NP and UNSAT is in coNP. Hardness follows from SAT-UNSAT [PW88] by mapping the first formula to an equivalent RX3C instance (via a standard NP-completeness reduction) and keeping the second formula unchanged.

RX3C-UNSAT
Input: An RX3C-instance $I_1 = (X_1, \mathcal{S})$ and a 3SAT-instance $I_2 = (X_2, \varphi)$ such that:

1. $|X_1| = 3\hat{n}_1$,
2. $\mathcal{S} = \{S_1, \ldots, S_{3\hat{n}_1}\}$ with $S_j \subseteq X_1$ and $|S_j| = 3$ for all $j \in [3\hat{n}_1]$,
3. each element of X_1 is contained in exactly three sets of $\mathcal{S}$, and
4. φ is a 3CNF formula containing only variables in X_2.

Question: Is it true that I_1 has an *exact cover* (i.e., a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that each $x \in X_1$ belongs to exactly one set of $\mathcal{S}'$) and I_2 has no satisfying truth assignment?

Let $I = (I_1, I_2)$ be an instance of RX3C-UNSAT, where $|X_1| = 3\hat{n}_1$, $|X_2| = \hat{n}_2$, and $|\varphi| = \hat{m}$, and each element of X_1 occurs in exactly three members in $\mathcal{S}$. W.l.o.g. assume that $\hat{m} + 2$ is divisible by 3 and that $\hat{m} > 1$.

Profile construction. Set $k \coloneqq 2\hat{n}_1 + \hat{n}_2 + 1$ and $L \coloneqq \frac{\hat{m}+2}{3}$.

- For each set $S_j \in \mathcal{S}$ create a *set1-alternative* s^1_j and a *set2-alternative* s^2_j; let $\mathcal{C}_{\mathcal{S}_1} \coloneqq \{\mathsf{s}^1_1, \ldots, \mathsf{s}^1_{3\hat{n}_1}\}$ and $\mathcal{C}_{\mathcal{S}_2} \coloneqq \{\mathsf{s}^2_1, \ldots, \mathsf{s}^2_{3\hat{n}_1}\}$.
- For each variable $x_i \in X_2$ create two *literal-alternatives* c_i and $\overline{c}_i$; let $\mathcal{C}_X \coloneqq \{c_1, \overline{c}_1, \ldots, c_{\hat{n}_2}, \overline{c}_{\hat{n}_2}\}$.
- Create a clause alternative c_φ and a dummy alternative c_d.
- For each element $x_i \in X_1$ create L *element1-voters* $v^1_{i,1}, \ldots, v^1_{i,L}$ and L *element2-voters* $v^2_{i,1}, \ldots, v^2_{i,L}$; let $\mathcal{V}^1 \coloneqq \{v^1_{1,1}, \ldots, v^1_{1,L}, \ldots, v^1_{3\hat{n}_1,1}, \ldots, v^1_{3\hat{n}_1,L}\}$, $\mathcal{V}^2 \coloneqq \{v^2_{1,1}, \ldots, v^2_{1,L}, \ldots, v^2_{3\hat{n}_1,1}, \ldots, v^2_{3\hat{n}_1,L}\}$. Their approvals are $\mathsf{A}(v^1_{i,\ell}) \coloneqq \{\mathsf{s}^1_j \mid x_i \in S_j\}$ and $\mathsf{A}(v^2_{i,\ell}) \coloneqq \{\mathsf{s}^2_j \mid x_i \in S_j\}$ for all $\ell \in [L]$.
- For each variable $x_i \in X_2$ create $3L$ *variable-voters* $v^x_{i,1}, \ldots, v^x_{i,3L}$; let $\mathcal{V}^X \coloneqq \{v^x_{1,1}, \ldots, v^x_{1,3L}, \ldots, v^x_{\hat{n}_2,1}, \ldots, v^x_{\hat{n}_2,3L}\}$. Their approvals are $\mathsf{A}(v^x_{i,\ell}) = \{c_i, \overline{c}_i\}$ for $\ell \in [3L]$.
- For each clause $C_j \in \varphi$ create a *clause-voter* v^φ_j; let $\mathcal{V}^\varphi \coloneqq \{v^\varphi_1, \ldots, v^\varphi_{\hat{m}}\}$. Their approvals are $\mathsf{A}(v^\varphi_j) \coloneqq \{c_i \mid x_i \in C_j\} \cup \{\overline{c}_i \mid \overline{x}_i \in C_j\} \cup \{c_\varphi\}$.
- Create a dummy voter $\mathsf{v_d}$ with $\mathsf{A}(\mathsf{v_d}) \coloneqq \{c_d\}$.

Thus the total number of voters is $|\mathcal{V}| = 2 \cdot (3\hat{n}_1) \cdot L + (3L)\hat{n}_2 + \hat{m} + 1 = (\hat{m}+2) \cdot (2\hat{n}_1 + \hat{n}_2) + \hat{m} + 1$.

The utility function $\boldsymbol{\alpha}$. Set $\boldsymbol{\alpha}(v) = 1$ for all $v \in \mathcal{V} \setminus \{\mathsf{v_d}\}$ and $\boldsymbol{\alpha}(\mathsf{v_d}) = 0$.

Correctness. We show that I is a yes-instance of RX3C-UNSAT if and only if $\boldsymbol{\alpha} \in \mathsf{TU}_{\mathrm{CC}}$-core.

For the "if" direction, we prove the contrapositive. Assume I is a no-instance of RX3C-UNSAT, meaning that I_1 has no exact cover or I_2 is satisfiable. We distinguish two cases.

Case 1: I_1 has no exact cover. We show that $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}) = \mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) < \sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = |\mathcal{V}| - 1$, hence $\boldsymbol{\alpha}$ is infeasible.

Let W be an arbitrary size-k committee. Define

$$W_1 \coloneqq W \cap \mathcal{C}_{\mathcal{S}_1}, \qquad W_2 \coloneqq W \cap \mathcal{C}_{\mathcal{S}_2}, \qquad W_3 \coloneqq W \setminus (\mathcal{C}_{\mathcal{S}_1} \cup \mathcal{C}_{\mathcal{S}_2}).$$

Since voters in $\mathcal{V}^1$ approve only alternatives in $\mathcal{C}_{\mathcal{S}_1}$, we have $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}^1, W) = \mathsf{sc}_{\mathrm{CC}}(\mathcal{V}^1, W_1)$, and analogously $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}^2, W) = \mathsf{sc}_{\mathrm{CC}}(\mathcal{V}^2, W_2)$ and $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V} \setminus (\mathcal{V}^1 \cup \mathcal{V}^2), W) = \mathsf{sc}_{\mathrm{CC}}(\mathcal{V} \setminus (\mathcal{V}^1 \cup \mathcal{V}^2), W_3)$.

We claim that if $|W_1| \leq \hat{n}_1$, then $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}^1, W_1) \leq |\mathcal{V}^1| - L$. Indeed, the sets corresponding to W_1 cover at most $3|W_1| \leq 3\hat{n}_1$ elements of X_1. Since $|X_1| = 3\hat{n}_1$ but I_1 has no exact cover, some element $x_i \in X_1$ is not covered by these sets. Then none of the L voters $v^1_{i,1}, \ldots, v^1_{i,L}$ is covered by W_1, and hence at least L voters in $\mathcal{V}^1$ are uncovered. Therefore $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}^1, W_1) \leq |\mathcal{V}^1| - L$. The same argument gives: if $|W_2| \leq \hat{n}_1$, then $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}^2, W_2) \leq |\mathcal{V}^2| - L$.

Next, note that the dummy voter $\mathsf{v_d}$ is covered by W_3 if and only if $c_d \in W_3$, but no other voter approves c_d. Moreover, the $\hat{n}_2$ variable voter groups are disjoint in the sense that for each $i \in [\hat{n}_2]$ the voters $\{v^x_{i,1}, \ldots, v^x_{i,3L}\}$ approve only $\{c_i, \bar{c}_i\}$, and different variables have disjoint approved sets. Thus, in order to cover all variable voters in $\mathcal{V}^X$, W_3 must contain at least one literal-alternative for each variable, i.e., at least $\hat{n}_2$ alternatives from $\mathcal{C}_X$. In particular, if $|W_3| \leq \hat{n}_2$, then W_3 cannot contain c_d and one literal for each variable at the same time, so at least one voter in $\mathcal{V} \setminus (\mathcal{V}^1 \cup \mathcal{V}^2)$ is uncovered. Hence

$$\text{if } |W_3| \leq \hat{n}_2, \text{ then } \mathsf{sc}_{\mathrm{CC}}(\mathcal{V} \setminus (\mathcal{V}^1 \cup \mathcal{V}^2), W_3) \leq |\mathcal{V} \setminus (\mathcal{V}^1 \cup \mathcal{V}^2)| - 1.$$

Now suppose, towards a contradiction, that $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}, W) \geq |\mathcal{V}| - 1$. Then the deficit of uncovered voters over all of $\mathcal{V}$ is at most 1. In particular, we cannot have $|W_1| \leq \hat{n}_1$ (otherwise $\mathcal{V}^1$ alone loses at least $L \geq 1$), so $|W_1| \geq \hat{n}_1 + 1$. Similarly, $|W_2| \geq \hat{n}_1 + 1$. Finally, we must have $|W_3| \geq \hat{n}_2$ (otherwise $\mathcal{V}^X$ loses at least $3L \geq 1$), and in fact $|W_3| \geq \hat{n}_2$ is necessary just to cover all variable voters. Therefore,

$$|W| = |W_1| + |W_2| + |W_3| \geq (\hat{n}_1 + 1) + (\hat{n}_1 + 1) + \hat{n}_2 = 2\hat{n}_1 + \hat{n}_2 + 2 > k,$$

a contradiction. Hence no size-k committee can cover at least $|\mathcal{V}| - 1$ voters, so $\mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) \leq |\mathcal{V}| - 2 < |\mathcal{V}| - 1 = \sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v)$ and $\boldsymbol{\alpha}$ is not feasible for $\mathcal{V}$.

Case 2: I_2 has a satisfying truth assignment and I_1 has an exact cover. We show $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}) = |\mathcal{V}| = \mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) > \sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = |\mathcal{V}| - 1$, hence $\boldsymbol{\alpha}$ is infeasible for $\mathcal{V}$ as well.

Let $\mathcal{S}'$ be an exact cover of I_1. Define

$$W_1 \coloneqq \{\mathsf{s}_j^1 \mid S_j \in \mathcal{S}'\}, \qquad W_2 \coloneqq \{\mathsf{s}_j^2 \mid S_j \in \mathcal{S}'\}.$$

Then $|W_1| = |W_2| = \hat{n}_1$ and W_1 (resp. W_2) covers all voters in $\mathcal{V}^1$ (resp. $\mathcal{V}^2$). Let σ be a satisfying assignment of φ, and define

$$W_3 \coloneqq \{c_i \mid \sigma(x_i) = \mathsf{T}\} \ \cup \ \{\bar{c}_i \mid \sigma(x_i) = \mathsf{F}\}.$$

Let $W \coloneqq W_1 \cup W_2 \cup W_3 \cup \{c_d\}$, then $|W| = \hat{n}_1 + \hat{n}_1 + \hat{n}_2 + 1 = k$ and W covers all voters in $\mathcal{V}$, hence $\mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) = |\mathcal{V}| > \sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v)$. This finishes the contrapositive and thus the "if" direction.

For the "only if" direction, assume that I is a yes-instance of RX3C-UNSAT, i.e., I_1 has an exact cover and φ is unsatisfiable. We first show feasibility of $\boldsymbol{\alpha}$, i.e., $\sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v) = \mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V})$.

Let $\mathcal{S}'$ be an exact cover of I_1 and define W_1, W_2 as above (so $|W_1| = |W_2| = \hat{n}_1$ and they cover $\mathcal{V}^1, \mathcal{V}^2$). Let

$$W_3 \coloneqq \{c_1, \ldots, c_{\hat{n}_2}\} \cup \{c_\varphi\}.$$

Then $|W_3| = \hat{n}_2 + 1$, and W_3 covers all variable voters (each group $\{v_{i,1}^x, \ldots, v_{i,3L}^x\}$ is covered by c_i) and all clause voters (every clause voter approves c_φ). Set

$$W^\star \coloneqq W_1 \cup W_2 \cup W_3.$$

Then $|W^\star| = \hat{n}_1 + \hat{n}_1 + (\hat{n}_2 + 1) = k$ and $W^\star$ covers all voters except $\mathsf{v_d}$. Hence $\mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) \geq |\mathcal{V}| - 1 = \sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v)$.

For the matching upper bound, let W be any size-k committee, and decompose it into W_1, W_2, W_3 as in Case 1. If $|W_1| < \hat{n}_1$ or $|W_2| < \hat{n}_1$, then $\mathcal{V}^1$ or $\mathcal{V}^2$ loses at least $L \geq 1$ voters, hence $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}, W) \leq |\mathcal{V}| - 1$. So suppose $|W_1| \geq \hat{n}_1$ and $|W_2| \geq \hat{n}_1$. Then $|W_3| \leq k - 2\hat{n}_1 = \hat{n}_2 + 1$. If $c_d \in W_3$, then W_3 contains at most $\hat{n}_2$ other alternatives from $\mathcal{C}_X \cup \{c_\varphi\}$. To cover all $\hat{n}_2$ variable voter groups, one needs at least $\hat{n}_2$ literal-alternatives, leaving no room for c_φ. In that situation, the literals in W_3 select at most one of $\{c_i, \bar{c}_i\}$ for each i (otherwise fewer variables are covered), and thus correspond to a truth assignment of X_2; since φ is unsatisfiable, at least one clause voter is not covered by these literals. Hence W fails to cover at least one voter among $\mathcal{V}^\varphi$. If instead $c_d \notin W_3$, then $\mathsf{v_d}$ is uncovered. In both cases, W leaves at least one voter uncovered and thus $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}, W) \leq |\mathcal{V}| - 1$. Together with the lower bound from $W^\star$, we obtain $\mathsf{TU}_{\mathrm{CC}}\text{-}\chi(\mathcal{V}) = |\mathcal{V}| - 1 = \sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v)$, so $\boldsymbol{\alpha}$ is feasible.

It remains to show that $\boldsymbol{\alpha}$ is not blocked.

Claim 5.37. *Assume I is a yes-instance. If there exists a blocking voter coalition $\mathcal{V}' \subseteq \mathcal{V}$, then there exists a blocking voter coalition $\hat{\mathcal{V}}$ with a witness $\hat{W}$ such that*

(i) $\mathsf{v_d} \in \hat{\mathcal{V}}$,

(ii) $\mathsf{sc}_{CC}(\hat{\mathcal{V}}, \hat{W}) = |\hat{\mathcal{V}}|$ *(i.e., every voter in $\hat{\mathcal{V}}$ is covered by $\hat{W}$),*

(iii) $\mathcal{V}^X \cup \mathcal{V}^1 \cup \mathcal{V}^2 \subseteq \hat{\mathcal{V}}$.

Proof. Let $\mathcal{V}'$ be blocking with witness W'.

Statement (i): If $\mathsf{v_d} \notin \mathcal{V}'$, then $\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) = |\mathcal{V}'|$, while always $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}', W') \leq |\mathcal{V}'|$, contradicting that $\mathcal{V}'$ blocks $\boldsymbol{\alpha}$. Hence $\mathsf{v_d} \in \mathcal{V}'$.

Statement (ii): By the first statement, since $\mathsf{v_d} \in \mathcal{V}'$, we have $\sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) = |\mathcal{V}'| - 1$. Therefore,

$$|\mathcal{V}'| - 1 \;=\; \sum_{v \in \mathcal{V}'} \boldsymbol{\alpha}(v) \;<\; \mathsf{sc}_{\mathrm{CC}}(\mathcal{V}', W') \;\leq\; |\mathcal{V}'|.$$

As the CC-score is integral, it follows that $\mathsf{sc}_{\mathrm{CC}}(\mathcal{V}', W') = |\mathcal{V}'|$, i.e., W' covers every voter in $\mathcal{V}'$.

Statement (iii): We now construct $(\hat{\mathcal{V}}, \hat{W})$ from $(\mathcal{V}', W')$ by successive operations that preserve blocking and maintain full coverage.

For the admissibility bound, we use the floor variant: a witness W' for $\mathcal{V}'$ must satisfy

$$|W'| \;\leq\; \left\lfloor \frac{k|\mathcal{V}'|}{|\mathcal{V}|} \right\rfloor.$$

We will repeatedly use that $|\mathcal{V}| = (\hat{m}+2)k - 1$ and $3L = \hat{m} + 2$, hence

$$\frac{|\mathcal{V}|}{k} = (\hat{m}+2) - \frac{1}{k} < 3L \tag{5.13}$$

Step 1: Close $\mathcal{V}'$ under copies (without changing the witness). We iteratively add missing copies of already present voters:

- If for some $i \in [\hat{n}_2]$ and some $\ell_1, \ell_2 \in [3L]$ we have $v^x_{i,\ell_1} \in \mathcal{V}'$ but $v^x_{i,\ell_2} \notin \mathcal{V}'$, then add v^x_{i,ℓ_2} to $\mathcal{V}'$.
- If for some $j \in [3\hat{n}_1]$ and some $\ell_1, \ell_2 \in [L]$ we have $v^1_{j,\ell_1} \in \mathcal{V}'$ but $v^1_{j,\ell_2} \notin \mathcal{V}'$, then add v^1_{j,ℓ_2} to $\mathcal{V}'$.
- If for some $j \in [3\hat{n}_1]$ and some $\ell_1, \ell_2 \in [L]$ we have $v^2_{j,\ell_1} \in \mathcal{V}'$ but $v^2_{j,\ell_2} \notin \mathcal{V}'$, then add v^2_{j,ℓ_2} to $\mathcal{V}'$.

Each such addition preserves blocking with the same witness W': Since each voter in $\mathcal{V}'$ is covered by W' (Statement (ii)) and the added voter is of the same type as a voter already in $\mathcal{V}'$, he is covered by W', and since $\boldsymbol{\alpha}$ assigns 1 to all non-dummy voters, both $\mathsf{sc}_{\mathrm{CC}}(\cdot, W')$ and $\sum \boldsymbol{\alpha}(\cdot)$ increase by exactly 1. The proportional bound $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ remains valid because the right-hand side increases when we add voters. After finitely many steps, we obtain a blocking coalition (still denoted $\mathcal{V}'$) such that for every $i \in [\hat{n}_2]$ either the entire voter group $\{v^x_{i,1}, \ldots, v^x_{i,3L}\}$ is contained in $\mathcal{V}'$ or it is disjoint from $\mathcal{V}'$, and analogously for each element voter group in $\mathcal{V}^1$ and $\mathcal{V}^2$.

Step 2: Add $\mathcal{V}^X$. If $\mathcal{V}^X \subseteq \mathcal{V}'$, we are done. Otherwise let $U \subseteq [\hat{n}_2]$ be the set of indices i whose entire block is missing, so $|\mathcal{V}^X \setminus \mathcal{V}'| = |U| \cdot 3L$. Define

$$\hat{\mathcal{V}} \;\coloneqq\; \mathcal{V}' \cup \bigcup_{i \in U} \{v^x_{i,1}, \ldots, v^x_{i,3L}\}.$$

Let $\hat{W}$ be obtained from W' by, for each $i \in U$ with $W' \cap \{c_i, \overline{c}_i\} = \emptyset$, adding one of $c_i, \overline{c}_i$. Then every newly added voter is covered by construction, and every voter in $\mathcal{V}'$ remains covered, so by the second statement,

$$\mathsf{sc}_{\mathrm{CC}}(\hat{\mathcal{V}}, \hat{W}) = |\hat{\mathcal{V}}|.$$

Moreover, $|\hat{W}| \leq |W'| + |U|$.

It remains to check admissibility for $(\hat{\mathcal{V}}, \hat{W})$. Since $|\hat{\mathcal{V}}| = |\mathcal{V}'| + |U| \cdot 3L$, we have

$$\frac{k|\hat{\mathcal{V}}|}{|\mathcal{V}|} = \frac{k|\mathcal{V}'|}{|\mathcal{V}|} + |U| \cdot \frac{k \cdot 3L}{|\mathcal{V}|}.$$

By (5.13), $\frac{k \cdot 3L}{|\mathcal{V}|} > 1$, hence

$$\left\lfloor \frac{k|\hat{\mathcal{V}}|}{|\mathcal{V}|} \right\rfloor \;\geq\; \left\lfloor \frac{k|\mathcal{V}'|}{|\mathcal{V}|} \right\rfloor + |U|.$$

Using $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ and $|\hat{W}| \leq |W'| + |U|$, we conclude $|\hat{W}| \leq \lfloor k|\hat{\mathcal{V}}|/|\mathcal{V}| \rfloor$. Thus $\hat{\mathcal{V}} = \mathcal{V}' \cup \mathcal{V}^X$ is blocking.

Step 3: Add $\mathcal{V}^1$ and $\mathcal{V}^2$ via an exact-cover exchange. We show the step for $\mathcal{V}^1$; the argument for $\mathcal{V}^2$ is identical. By Step 1, we may assume that for each $j \in [3\hat{n}_1]$ either the whole voter group $\{v^1_{j,1}, \ldots, v^1_{j,L}\}$ is contained in $\mathcal{V}'$ or disjoint from $\mathcal{V}'$.

Let W' witness that $\mathcal{V}'$ is blocking. Set g_1 such that $|\mathcal{V}' \cap \mathcal{V}^1| = g_1 \cdot L$ (number of present element-blocks), and let $g_2 \coloneqq |W' \cap \mathcal{C}_{\mathcal{S}_1}|$. By Statement (ii), every voter in $\mathcal{V}' \cap \mathcal{V}^1$ is covered by W', and only alternatives in $\mathcal{C}_{\mathcal{S}_1}$ can cover them. Each alternative in $\mathcal{C}_{\mathcal{S}_1}$ corresponds to a 3-set and hence can cover voters from at most three element-voter groups. Therefore $g_1 \leq 3g_2$.

Let $W_x \subseteq \mathcal{C}_{\mathcal{S}_1}$ be the $\hat{n}_1$ set1-alternatives corresponding to an exact cover of I_1. Define

$$\hat{\mathcal{V}} \;\coloneqq\; \mathcal{V}' \cup \mathcal{V}^1, \qquad \hat{W} \;\coloneqq\; (W' \setminus \mathcal{C}_{\mathcal{S}_1}) \cup W_x.$$

Then $\hat{W}$ covers all voters in $\mathcal{V}^1$ (exact cover) and does not affect coverage of voters outside $\mathcal{V}^1$, since no voter outside $\mathcal{V}^1$ approves alternatives in $\mathcal{C}_{\mathcal{S}_1}$. Hence $\mathsf{sc}_{\mathrm{CC}}(\hat{\mathcal{V}}, \hat{W}) = |\hat{\mathcal{V}}|$.

For admissibility, note that $|\hat{W}| = |W'| - g_2 + \hat{n}_1$ and $|\hat{\mathcal{V}}| = |\mathcal{V}'| + (3\hat{n}_1 - g_1)L$. Using $g_1 \leq 3g_2$ we obtain

$$|\hat{\mathcal{V}}| = |\mathcal{V}'| + (3\hat{n}_1 - g_1)L \;\geq\; |\mathcal{V}'| + 3(\hat{n}_1 - g_2)L.$$

Now apply (5.13) with $|U| = \hat{n}_1 - g_2$: adding $3L$ voters increases the floor budget by at least 1, hence adding $3(\hat{n}_1 - g_2)L$ voters increases it by at least $\hat{n}_1 - g_2$. Formally,

$$\left\lfloor \frac{k|\hat{\mathcal{V}}|}{|\mathcal{V}|} \right\rfloor \;\geq\; \left\lfloor \frac{k|\mathcal{V}'|}{|\mathcal{V}|} \right\rfloor + (\hat{n}_1 - g_2).$$

Using $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$ and $|\hat{W}| = |W'| + (\hat{n}_1 - g_2)$, we obtain $|\hat{W}| \leq \lfloor k|\hat{\mathcal{V}}|/|\mathcal{V}| \rfloor$. Thus $\mathcal{V}' \cup \mathcal{V}^1$ is blocking.

Analogously, we can show that $\mathcal{V}' \cup \mathcal{V}^2$ is blocking with an admissible committee.

(end of the proof of Claim 5.37 $\diamond$)

Now suppose, towards a contradiction, that a blocking coalition exists. By Claim 5.37 there exists a blocking coalition $\hat{\mathcal{V}}$ containing $\mathsf{v_d}$ and all voters in $\mathcal{V}^X \cup \mathcal{V}^1 \cup \mathcal{V}^2$, with witness $\hat{W}$ covering all voters in $\hat{\mathcal{V}}$. Then $\hat{W}$ must contain at least $\hat{n}_1$ alternatives from $\mathcal{C}_{\mathcal{S}_1}$ (to cover $\mathcal{V}^1$), $\hat{n}_1$ from $\mathcal{C}_{\mathcal{S}_2}$ (to cover $\mathcal{V}^2$), $\hat{n}_2$ from $\mathcal{C}_X$ (to cover $\mathcal{V}^X$), and c_d (to cover $\mathsf{v_d}$). Thus $|\hat{W}| \geq 2\hat{n}_1 + \hat{n}_2 + 1 = k$. But also $|\hat{W}| \leq k \cdot |\hat{\mathcal{V}}|/|\mathcal{V}| \leq k$, hence $|\hat{W}| = k$, which implies $|\hat{\mathcal{V}}| = |\mathcal{V}|$ and thus $\hat{\mathcal{V}} = \mathcal{V}$. Therefore $\mathcal{V}$ would be blocking, contradicting $\mathsf{TU}_{\mathrm{CC}}$-$\chi(\mathcal{V}) = \sum_{v \in \mathcal{V}} \boldsymbol{\alpha}(v)$ shown above. Hence, no blocking coalition exists and $\boldsymbol{\alpha} \in \mathsf{TU}_{\mathrm{CC}}$-core. □

Finally, the NP-hardness for PAV comes from the fact that computing a winning committee under PAV is NP-hard [AGG+15].

Proposition 5.38. *PAV-*TU*-CV is* NP*-hard.*

Proof. We reduce from RX3C. Let $I = (X, \mathcal{S})$ be an RX3C instance with $X = \{x_1, \ldots, x_{3\hat{n}}\}$; the definition of RX3C can be found in the proof of Theorem 5.26. We create an instance $I' = (\mathcal{P}, k, \boldsymbol{\alpha})$ of PAV-TU-CV as follows.

Profile construction. Let $k \coloneqq \hat{n}$.

- For each set $S_j \in \mathcal{S}$, create an alternative c_j.
- For each element $x_i \in X$, create a voter v_i. Their approvals are $\mathsf{A}(v_i) \coloneqq \{c_j \mid x_i \in S_j\}$.

Thus the total number of voters is $|\mathcal{V}| = 3\hat{n}$.

The utility function $\boldsymbol{\alpha}$. $\boldsymbol{\alpha}(v_i) = 1$ for all $v_i \in \mathcal{V}$.

Correctness. We show that I is a yes-instance of RX3C if and only if $\boldsymbol{\alpha}$ is in the $\mathsf{TU}_{\mathrm{PAV}}$-core.

For the "if" direction, suppose $\boldsymbol{\alpha} \in \mathsf{TU}_{\mathrm{PAV}}$-core. By definition, there must exist a winning committee that realizes $\boldsymbol{\alpha}$. Let W be such a committee, i.e., $|W| = k = \hat{n}$ and $\mathsf{sc}_{\mathrm{PAV}}(\mathcal{V}, W) = \sum_{i=1}^{3\hat{n}} \boldsymbol{\alpha}(v_i) = 3\hat{n} = |\mathcal{V}|$. As each alternative $c_j \in \mathcal{C}$ is approved by exactly three voters, it holds that each voter must approve exactly one alternative in W; otherwise the score would be less than $|\mathcal{V}|$ due to the definition of the PAV-score. Then, it must also hold that the sets corresponding to alternatives in W form an exact cover: Each element is covered by them exactly once. This concludes the "if" direction.

For the "only if" direction, suppose that I is a yes-instance and let $\mathcal{S}'$ be an exact cover for I. We show that $\boldsymbol{\alpha}$ is realized by a size-k committee and no coalition (including the grand coalition) should be able to block $\boldsymbol{\alpha}$.

For the realizability, define $W \coloneqq \{c_j \mid S_j \in \mathcal{S}'\}$. Then, $|W| = \hat{n}$. Moreover, $\mathsf{sc}_{\mathrm{PAV}}(\mathcal{V}, W) = |\mathcal{V}|$ since $\mathcal{S}'$ covers each element exactly once.

It remains to show that no coalition is blocking. Suppose for the sake of contradiction, that $\mathcal{V}' \subseteq \mathcal{V}$ is a coalition that is $\mathsf{TU}_{\mathrm{PAV}}$-blocking $\boldsymbol{\alpha}$ and W' is a witness. By the admissibility of W', we infer

$$|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor = \lfloor \frac{|\mathcal{V}'|}{3} \rfloor \leq \frac{|\mathcal{V}'|}{3}. \tag{5.14}$$

By the score, we infer $\mathsf{sc}_{\mathrm{PAV}}(\mathcal{V}', W') > \sum_{v_i \in \mathcal{V}'} \boldsymbol{\alpha}(v_i) = |\mathcal{V}'|$. Since each alternative is approved by exactly three voters, it follows that $3|W'| \geq \mathsf{sc}_{\mathrm{PAV}}(\mathcal{V}', W') > |\mathcal{V}'|$, a contradiction to (5.14). Therefore $\boldsymbol{\alpha} \in \mathsf{TU}_{\mathrm{PAV}}$-core. This concludes the proof. □

5.4.2 Complexity for the NTU Model

We now consider the NTU model, where coalitions cannot redistribute utility. Here, blocking requires a *component-wise* strict improvement for every coalition member, which makes core membership verification inherently more combinatorial: A witness must specify an admissible committee whose induced per-voter scores dominate the proposed utilities on the entire coalition.

By Theorem 5.22, we immediately obtain the following polynomial result.

Theorem 5.39. *CC-NTU-CE is in* P.

For the core membership problem, we show a somewhat interesting contrast to the TU model. It becomes DP-complete for AV and hence SAV and PAV (see Observation 5.18),

but is NP-complete for CC. We prove the DP-hardness by a reduction from the following DP-hard problem, denoted RX3C-UNRX3C.[12]

RX3C-UNRX3C
Input: Two RX3C instances $I_1 = (X_1, \mathcal{S}_1)$ and $I_2 = (X_2, \mathcal{S}_2)$ such that for a common $\hat{n} \in \mathbb{N}$ and each $z \in [2]$:

1. $|X_z| = 3\hat{n}$,
2. $\mathcal{S}_z = \{S_1^z, \dots, S_{3\hat{n}}^z\}$ with $S_j^z \subseteq X_z$ and $|S_j^z| = 3$ for all $j \in [3\hat{n}]$, and
3. each element of X_z is contained in exactly three sets of $\mathcal{S}_z$.

Question: Is it true that I_1 has an *exact cover* (i.e., a subcollection $\mathcal{S}_1' \subseteq \mathcal{S}_1$ of cardinality $\hat{n}$ such that each $x \in X_1$ belongs to exactly one set of $\mathcal{S}_1'$) while I_2 does not?

Our reduction extends a hardness reduction by Brill et al. [BGP+24], which shows that verifying whether a committee is in the core is coNP-hard.

Theorem 5.40. *AV-*NTU*-CV and hence SAV-*NTU*-CV and PAV-*NTU*-CV are* DP*-complete.*

Proof. DP-membership follows from Theorem 5.24.2. By Observation 5.18, it suffices to show DP-hardness for AV-NTU-CV. As mentioned, we reduce from RX3C-UNRX3C.

Let $I = (I_1, I_2)$ be an instance of RX3C-UNRX3C, where for $z \in [2]$, $I_z = (X_z, \mathcal{S}_z)$, $|X_z| = 3\hat{n}$, $\mathcal{S}_z = \{S_1^z, \dots, S_{3\hat{n}}^z\}$, each $|S_j^z| = 3$, and each element of X_z occurs in exactly three sets. We construct an instance $I' = (\mathcal{P}, k, \boldsymbol{\alpha})$ of AV-NTU-CV.

Profile construction. Set $k \coloneqq 2\hat{n}$.

- For each $S_j^1 \in \mathcal{S}_1$ create a *set1-alternative* s_j^1 and let $\mathcal{C}_{\mathcal{S}_1} \coloneqq \{\mathsf{s}_1^1, \dots, \mathsf{s}_{3\hat{n}}^1\}$.
- For each $S_j^2 \in \mathcal{S}_2$ create a *set2-alternative* s_j^2 and let $\mathcal{C}_{\mathcal{S}_2} \coloneqq \{\mathsf{s}_1^2, \dots, \mathsf{s}_{3\hat{n}}^2\}$.
- Create private alternatives $\mathcal{C}_P \coloneqq \{p_1, \dots, p_{\hat{n}-1}\}$ and one dummy alternative c_d.
- For each $x_i^1 \in X_1$ create an *element1-voter* v_i^1 with approvals $\mathsf{A}(v_i^1) \coloneqq \{\mathsf{s}_j^1 \mid x_i^1 \in S_j^1\}$, and let $\mathcal{V}_{X_1} \coloneqq \{v_i^1 \mid x_i^1 \in X_1\}$.

[12] RX3C-UNRX3C is DP-hard: Since the underlying RX3C (short for Restricted Exact Cover by 3-Sets) decision problem (asking whether I_1 has an exact cover) is NP-complete, there exists a polynomial-time reduction r from SAT to RX3C. Hence $(\varphi_1, \varphi_2) \mapsto (r(\varphi_1), r(\varphi_2))$ yields a polynomial-time reduction from the DP-complete problem SAT-UNSAT [PW88] to RX3C-UNRX3C.

- For each $x_i^2 \in X_2$ create an *element2-voter* v_i^2 with approvals $\mathsf{A}(v_i^2) \coloneqq \{\mathsf{s}_j^2 \mid x_i^2 \in S_j^2\}$, and let $\mathcal{V}_{X_2} \coloneqq \{v_i^2 \mid x_i^2 \in X_2\}$.
- Add two dummy voters: d_1 with $\mathsf{A}(d_1) = \mathcal{C}_{\mathcal{S}_1}$ and d_2 with $\mathsf{A}(d_2) = \mathcal{C}_{\mathcal{S}_2} \cup \mathcal{C}_P$.

Thus $\mathcal{C} = \mathcal{C}_{\mathcal{S}_1} \cup \mathcal{C}_{\mathcal{S}_2} \cup \mathcal{C}_P \cup \{c_d\}$ and $|\mathcal{V}| = |\mathcal{V}_{X_1}| + |\mathcal{V}_{X_2}| + 2 = 6\hat{n} + 2$.

The utility function $\boldsymbol{\alpha}$. Set $\boldsymbol{\alpha}(v_i^1) = 1$ for all $v_i^1 \in \mathcal{V}_{X_1}$, $\boldsymbol{\alpha}(v_i^2) = 0$ for all $v_i^2 \in \mathcal{V}_{X_2}$, and $\boldsymbol{\alpha}(d_1) = \hat{n}$, $\boldsymbol{\alpha}(d_2) = \hat{n} - 1$.

Correctness. We show that I is a yes-instance of RX3C-UNRX3C if and only if $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}$-core.

For the "if" direction, assume $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}$-core. Then, $\boldsymbol{\alpha}$ is feasible for $\mathcal{V}$ (i.e., $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}$-$\mu(\mathcal{V})$), so there exists a size-k committee W with $\mathsf{sc}_{\mathrm{AV}}(v, W) = \boldsymbol{\alpha}(v)$ for all $v \in \mathcal{V}$. From $\boldsymbol{\alpha}(d_1) = \hat{n}$ and $\mathsf{A}(d_1) = \mathcal{C}_{\mathcal{S}_1}$ it follows that $|W \cap \mathcal{C}_{\mathcal{S}_1}| = \hat{n}$. Moreover, for each element1-voter v_i^1 we have $\boldsymbol{\alpha}(v_i^1) = 1$ and $\mathsf{A}(v_i^1) \subseteq \mathcal{C}_{\mathcal{S}_1}$, hence $|W \cap \mathsf{A}(v_i^1)| = 1$. Therefore, the $\hat{n}$ chosen set1-alternatives cover every element of X_1 exactly once, so I_1 has an exact cover.

It remains to show that I_2 has no exact cover. Suppose, for contradiction, that I_2 admits an exact cover $\mathcal{S}' \subseteq \mathcal{S}_2$ of size $\hat{n}$ and set $W' \coloneqq \{\mathsf{s}_j^2 \mid S_j^2 \in \mathcal{S}'\}$, and $\mathcal{V}' = \mathcal{V}_{X_2} \cup \{d_2\}$. We claim that $\mathcal{V}'$ is blocking with witness W'. Clearly, $|W'| = \hat{n}$. Each element2-voter v_i^2 satisfies $\mathsf{sc}_{\mathrm{AV}}(v_i^2, W') = 1 > \boldsymbol{\alpha}(v_i^2) = 0$, and also $\mathsf{sc}_{\mathrm{AV}}(d_2, W') = \hat{n} > \boldsymbol{\alpha}(d_2) = \hat{n} - 1$. Since $|W'| = \hat{n} = k \cdot (3\hat{n}+1)/(6\hat{n}+2) = \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$, it is admissible for $\mathcal{V}_{X_2} \cup \{d_2\}$. So, $\mathcal{V}'$ is $\mathsf{NTU}_{\mathrm{AV}}$-blocking against $\boldsymbol{\alpha}$, contradicting $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}$-core. Hence I_2 is a no-instance, and I is a yes-instance of RX3C-UNRX3C.

For the "only if" direction, assume I is a yes-instance, i.e., I_1 has an exact cover and I_2 has none. We show $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}$-core by establishing feasibility and ruling out blocking coalitions.

Let $\mathcal{S}' \subseteq \mathcal{S}_1$ be an exact cover of size $\hat{n}$ and define $W_S \coloneqq \{\mathsf{s}_j^1 \mid S_j^1 \in \mathcal{S}'\}$ and $W \coloneqq W_S \cup \mathcal{C}_P \cup \{c_d\}$. Then $|W| = \hat{n} + (\hat{n} - 1) + 1 = 2\hat{n} = k$. By construction, $\mathsf{sc}_{\mathrm{AV}}(v_i^1, W) = 1$, $\mathsf{sc}_{\mathrm{AV}}(d_1, W) = \hat{n}$, $\mathsf{sc}_{\mathrm{AV}}(v_i^2, W) = 0$, and $\mathsf{sc}_{\mathrm{AV}}(d_2, W) = \hat{n} - 1$, hence $\mathsf{sc}_{\mathrm{AV}}(v, W) = \boldsymbol{\alpha}(v)$ for all $v \in \mathcal{V}$. Thus $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}$-$\mu(\mathcal{V})$.

Now we show that no coalition is blocking. Suppose for contradiction that some coalition $\mathcal{V}' \subseteq \mathcal{V}$ is $\mathsf{NTU}_{\mathrm{AV}}$-blocking against $\boldsymbol{\alpha}$ with witness committee W'.

Claim 5.41. *If $\mathcal{V}'$ $\mathsf{NTU}_{\mathrm{AV}}$-blocks $\boldsymbol{\alpha}$, then $\mathcal{V}' \setminus (\mathcal{V}_{X_1} \cup \{d_1\})$ $\mathsf{NTU}_{\mathrm{AV}}$-blocks $\boldsymbol{\alpha}$ as well.*

Proof. Let $\mathcal{V}'$ be a blocking coalition with witness committee W'. Write $W_1 \coloneqq W' \cap \mathcal{C}_{\mathcal{S}_1}$ and $\hat{W} \coloneqq W' \setminus \mathcal{C}_{\mathcal{S}_1}$, and define $\hat{\mathcal{V}} \coloneqq \mathcal{V}' \setminus (\mathcal{V}_{X_1} \cup \{d_1\})$. Every voter in $\hat{\mathcal{V}}$ approves no set1-alternatives, hence $\mathsf{sc}_{\mathrm{AV}}(v, \hat{W}) = \mathsf{sc}_{\mathrm{AV}}(v, W')$ for all $v \in \hat{\mathcal{V}}$, so strict improvement over $\boldsymbol{\alpha}$ is preserved for $\hat{\mathcal{V}}$.

It remains to show that $\hat{\mathcal{V}} \neq \emptyset$ and that $\hat{W}$ is admissible for $\hat{\mathcal{V}}$. We distinguish two cases.

Case A: $d_1 \in \mathcal{V}'$**.** Blocking requires $\mathsf{sc}_{\mathrm{AV}}(d_1, W') > \hat{n} = \boldsymbol{\alpha}(d_1)$, hence $|W_1| = |W' \cap \mathcal{C}_{\mathcal{S}_1}| \geq \hat{n}+1$. Since W' is admissible for $\mathcal{V}'$, we have $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor$, implying $|\mathcal{V}'|/|W'| \geq |\mathcal{V}|/k = (3\hat{n}+1)/\hat{n}$. Together with $|W'| \geq |W_1| \geq \hat{n}+1$, this yields $|\mathcal{V}'| > (3\hat{n}+1)$, and thus $\hat{\mathcal{V}} = \mathcal{V}' \setminus (\mathcal{V}_{X_1} \cup \{d_1\}) \neq \emptyset$. To see that $\hat{W}$ is admissible for $\hat{\mathcal{V}}$, write $q \coloneqq k/|\mathcal{V}| = \hat{n}/(3\hat{n}+1)$ and note that $|\hat{W}| = |W'| - |W_1|$ and $|\hat{\mathcal{V}}| \geq |\mathcal{V}'| - (3\hat{n}+1)$. Using $|W'| \leq q|\mathcal{V}'|$ (since W' is admissible for $\mathcal{V}'$) and $|W_1| \geq \hat{n}+1 \geq q(3\hat{n}+1)+1$, we obtain $|\hat{W}| \leq q|\mathcal{V}'| - q(3\hat{n}+1) - 1 \leq q|\hat{\mathcal{V}}| - 1 \leq \lfloor \frac{|\hat{\mathcal{V}}|k}{|\mathcal{V}|} \rfloor$, so $\hat{W}$ is admissible for $\hat{\mathcal{V}}$.

Case B: $d_1 \notin \mathcal{V}'$**.** Let $T \coloneqq \mathcal{V}' \cap \mathcal{V}_{X_1}$. Assume $T \neq \emptyset$ as otherwise $\hat{\mathcal{V}} = \mathcal{V}'$ and $\hat{W} = W'$, so the claim is immediate.

Since $\boldsymbol{\alpha}(v_i^1) = 1$ for all $v_i^1 \in \mathcal{V}_{X_1}$ and $\mathcal{V}'$ $\mathsf{NTU}_{\mathrm{AV}}$-blocks $\boldsymbol{\alpha}$, we have $\mathsf{sc}_{\mathrm{AV}}(v_i^1, W') > \boldsymbol{\alpha}(v_i^1) = 1$ for every $v_i^1 \in T$, and hence $\mathsf{sc}_{\mathrm{AV}}(v_i^1, W') \geq 2$. As v_i^1 approves only set1-alternatives, this implies $|W_1 \cap \mathsf{A}(v_i^1)| \geq 2$ for all $v_i^1 \in T$. Double counting approval incidences between T and W_1 yields $2|T| \leq \sum_{v_i^1 \in T} |W_1 \cap \mathsf{A}(v_i^1)| \leq 3|W_1|$, because each set1-alternative is approved by exactly three voters in $\mathcal{V}_{X_1}$. Thus,

$$|W_1| \geq \frac{2}{3}|T|. \tag{5.15}$$

Similarly to Case A, we show that $\hat{\mathcal{V}}$ is non-empty and $\hat{W}$ is admissible for $\hat{\mathcal{V}}$. Suppose towards a contradiction that $\hat{\mathcal{V}} = \emptyset$. Then $\mathcal{V}' = T \subseteq \mathcal{V}_{X_1}$. Hence,

$$|W'| \geq |W_1| \overset{(5.15)}{\geq} \frac{2}{3}|T| = \frac{2}{3}|\mathcal{V}'|. \tag{5.16}$$

On the other hand, admissibility of W' for $\mathcal{V}'$ gives $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor = \lfloor \hat{n}/(3\hat{n}+1)\,|\mathcal{V}'| \rfloor < \frac{1}{3}|\mathcal{V}'|$, a contradiction to (5.16). Therefore $\hat{\mathcal{V}} \neq \emptyset$.

Now, we consider admissibility of $\hat{W}$. Let $q \coloneqq k/|\mathcal{V}| = \hat{n}/(3\hat{n}+1)$. From admissibility of W' for $\mathcal{V}'$ we have $|W'| \leq q|\mathcal{V}'|$. Moreover, $|\hat{W}| = |W'| - |W_1|$ and $|\hat{\mathcal{V}}| = |\mathcal{V}'| - |T|$. Using (5.15), $q < \frac{1}{3}$, and $|T| \geq 1$, we obtain $|W_1| > q|T|$ and thus $|\hat{W}| < q|\mathcal{V}'| - q|T| = q|\hat{\mathcal{V}}| = \frac{|\hat{\mathcal{V}}|k}{|\mathcal{V}|}$. Since $|\hat{W}|$ is integral, this implies that $|\hat{W}| \leq \lfloor \frac{|\hat{\mathcal{V}}|k}{|\mathcal{V}|} \rfloor$, so $\hat{W}$ is admissible for $\hat{\mathcal{V}}$.

In all cases, $\hat{\mathcal{V}}$ is non-empty and $\mathsf{NTU}_{\mathrm{AV}}$-blocks $\boldsymbol{\alpha}$ witnessed by $\hat{W}$, and $\hat{\mathcal{V}} \cap (\mathcal{V}_{X_1} \cup \{d_1\}) = \emptyset$ by construction. (end of the proof of Claim 5.41 $\diamond$)

Claim 5.42. *If $\mathcal{V}'$ $\mathsf{NTU}_{\mathrm{AV}}$-blocks $\boldsymbol{\alpha}$ and $\mathcal{V}' \cap (\mathcal{V}_{X_1} \cup \{d_1\}) = \emptyset$, then $d_2 \in \mathcal{V}'$.*

Proof. Assume $\mathcal{V}'$ $\mathsf{NTU}_{\mathrm{AV}}$-blocks $\boldsymbol{\alpha}$ and $\mathcal{V}' \cap (\mathcal{V}_{X_1} \cup \{d_1\}) = \emptyset$. Suppose for the sake of contradiction that $d_2 \notin \mathcal{V}'$. Then $\mathcal{V}' \subseteq \mathcal{V}_{X_2}$ and, since $\boldsymbol{\alpha}(v_i^2) = 0$, blocking implies $\mathsf{sc}_{\mathrm{AV}}(v_i^2, W') \geq 1$ for all $v_i^2 \in \mathcal{V}'$. Each set2-alternative is approved by exactly three element2-voters, so $|W'| \geq |\mathcal{V}'|/3$. But admissibility implies $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor = \lfloor \frac{\hat{n}}{3\hat{n}+1}|\mathcal{V}'| \rfloor < \frac{|\mathcal{V}'|}{3}$, a contradiction. Hence $d_2 \in \mathcal{V}'$. (end of the proof of Claim 5.42 $\diamond$)

By Claim 5.41 we may assume $\mathcal{V}' \cap (\mathcal{V}_{X_1} \cup \{d_1\}) = \emptyset$, hence $\mathcal{V}' \subseteq \mathcal{V}_{X_2} \cup \{d_2\}$; note that this implies that $|\mathcal{V}'| \leq 3\hat{n} + 1$. By Claim 5.42, we have $d_2 \in \mathcal{V}'$.

Since $\boldsymbol{\alpha}(d_2) = \hat{n} - 1$, blocking implies $\mathsf{sc}_{\mathrm{AV}}(d_2, W') \geq \hat{n}$. As d_2 approves exactly $\mathcal{C}_{\mathcal{S}_2} \cup \mathcal{C}_P$, this forces $|W'| \geq \hat{n}$. Moreover, admissibility gives $|W'| \leq \lfloor \frac{|\mathcal{V}'|k}{|\mathcal{V}|} \rfloor \leq k \cdot |\mathcal{V}'|/|\mathcal{V}| \leq k \cdot (3\hat{n}+1)/(6\hat{n}+2) = \hat{n}$, so $|W'| = \hat{n}$ and thus $|\mathcal{V}'| = 3\hat{n} + 1$, i.e., $\mathcal{V}' = \mathcal{V}_{X_2} \cup \{d_2\}$.

Now, for each $v_i^2 \in \mathcal{V}_{X_2}$ blocking requires $\mathsf{sc}_{\mathrm{AV}}(v_i^2, W') \geq 1$. Each alternative in $\mathcal{C}_{\mathcal{S}_2}$ corresponds to a 3-set, hence is approved by exactly three element2-voters. Since $|W'| = \hat{n}$, W' can "cover" at most $3\hat{n}$ element2-voters, and to improve all $3\hat{n}$ element2-voters, it must cover all of them. This implies that the $\hat{n}$ selected 3-sets cover X_2 and, since their total size is $3\hat{n}$, they are pairwise disjoint; hence they form an exact cover for I_2, contradicting that I_2 is a no-instance. Thus, no blocking coalition exists and $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{AV}}$-core.

This establishes DP-hardness for AV-NTU-CV. The statement for SAV and PAV follows from Observation 5.18. □

Finally, we show an arguably interesting phenomenon: Verifying CC-core membership is NP-complete. The NP-hardness follows from the fact that computing a CC-winning committee is known to be NP-hard. Membership in NP follows because a committee W witnessing feasibility of $\boldsymbol{\alpha}$ serves as a polynomial-size certificate; given W, core membership can be checked in polynomial time by verifying JR; see Proposition 5.20.

Theorem 5.43. *CC-*NTU*-CV is* NP*-complete.*

Proof. We show NP-hardness by a reduction from the NP-complete problem Set Cover; see Section 2.7.

Given an instance $I = ([\hat{n}], \mathcal{S}, h)$ we construct an instance $I' = (\mathcal{P}, k, \boldsymbol{\alpha})$ of CC-NTU-CV as follows. Let $\mathcal{V} = \{v_1, \ldots, v_{\hat{n}}\}$ and $\mathcal{C} = \{c_1, \ldots, c_{\hat{m}}\}$, where c_j corresponds to S_j. For each voter $v_i \in \mathcal{V}$, set $\mathsf{A}(v_i) = \{c_j \mid i \in S_j\}$. Finally, set $k = h$ and define $\boldsymbol{\alpha}(v_i) = 1$ for all $v_i \in \mathcal{V}$.

We claim that I is a yes-instance if and only if I' is a yes-instance.

Assume that I is a yes-instance, and let $S_{j_1}, \ldots, S_{j_h}$ be a set cover. Let $W = \{c_{j_1}, \ldots, c_{j_h}\}$. Then for every $i \in [\hat{n}]$ we have $W \cap \mathsf{A}_i \neq \emptyset$, hence $\mathsf{sc}_{\mathrm{CC}}(v_i, W) = 1 = \boldsymbol{\alpha}(v_i)$. Since no voter can obtain utility strictly larger than 1 under CC, the profile $\boldsymbol{\alpha}$ cannot be blocked. Thus $\boldsymbol{\alpha} \in \mathsf{NTU}_{\mathrm{CC}}$-core and I' is a yes-instance.

Assume that I' is a yes-instance, and let W be a committee with $|W| = k = h$ such that $\mathsf{sc}_{\mathrm{CC}}(v_i, W) = \boldsymbol{\alpha}(v_i) = 1$ for all $v_i \in \mathcal{V}$. Equivalently, $W \cap \mathsf{A}_i \neq \emptyset$ for all $i \in [\hat{n}]$. Therefore, the h sets corresponding to the alternatives in W cover U, and I is a yes-instance.

For NP-membership, we use a committee W as a certificate. Let W be a committee with $|W| = k$. We can verify in polynomial time that $\boldsymbol{\alpha}(v_i) = \mathsf{sc}_{\mathrm{CC}}(v_i, W)$ for all $v_i \in \mathcal{V}$. By Proposition 5.20, it then suffices to check whether W satisfies JR, which can be done as

follows. Let $\mathcal{V}_0 \coloneqq \{v_i \in \mathcal{V} \mid W \cap \mathsf{A}_i = \emptyset\}$ be the set of unrepresented voters. Then W violates JR if and only if there exists an alternative $c \in \mathcal{C}$ such that $|\mathsf{V}(c) \cap \mathcal{V}_0| \geq \lceil |\mathcal{V}|/k \rceil$; recall that $\mathsf{V}(c) \coloneqq \{v_i \in \mathcal{V} \mid c \in \mathsf{A}_i\}$. This condition can be checked in polynomial time by scanning all alternatives and counting approvals inside $\mathcal{V}_0$. Hence CC-NTU-CV lies in NP, completing the proof. □

Note that CC-NTU-CV is polynomial-time solvable when a witnessing committee W is included in the input, since the task reduces to verifying $\boldsymbol{\alpha}(v_i) = \mathsf{sc}_{\mathrm{CC}}(v_i, W)$ for all v_i and checking JR for W. We conclude by discussing the broader implications and open directions.

5.5 Conclusion

We introduced *multi-winner games*, a cooperative-game framework for multi-winner voting in which coalitions are constrained by proportional entitlements and outcomes are rule-induced utility functions. The framework supports both NTU and TU interpretations, separating instability that is inherent to the committee feasibility constraints from instability that disappears once coalitions can redistribute utility internally.

Our results demonstrate that transferability fundamentally reshapes the stability landscape in predictable ways. The stark contrast—AV and SAV are well-behaved under both TU and NTU, while CC and PAV are well-behaved only under NTU—reflects a structural principle identified in the introduction. Under TU, feasibility ties to winning committees, making rule-specific properties actionable; under NTU, this link is severed. The Bondareva–Shapley approach [Bon63, Sha67] applies broadly, to any rule where the total committee score is the sum of individual voter scores and each alternative's contribution is independent of which others are selected. For any such totally separable rule, fractional committees cannot outperform integral ones, so the TU-core is always non-empty. If proportional entitlements are transferable, such rules thus provide robust stability. If entitlements are rigid, rule choice matters less than utility structure: CC's binary scores enable the JR characterization, while richer score ranges resisted the techniques available here.

On the computational side, we studied Core-Stability Existence (CE) and Core-Stability Verification (CV) for both utility models and four rules. Even when stable outcomes exist and can be constructed, CV can be challenging because blocking quantifies over exponentially many coalitions and admissible committees. Our complexity classifications delineate tractable cases (notably AV-TU-CE, SAV-TU-CE, and CC-NTU-CE) from settings where CE and/or CV become computationally hard.

Two concrete open problems warrant attention. First, the TU model suggests connections to market-based interpretations of proportional entitlements (e.g., prices or transferable compensation) and raises the question of which notions of fairness or distributional constraints are compatible with core stability. Second, it would be useful to develop approximation and relaxation frameworks that interpolate between NTU and TU (e.g.,

limited transfers, bounded side payments, or randomized outcomes) and to understand whether they recover existence or improve computational tractability. There are multiple ways to continue research on this topic: Extending the framework beyond approval utilities—for instance, to participatory budgeting or linear preferences—may help identify the general principles determining when proportional entitlements yield stable outcomes. Another extension would be to assign weights to the voters in terms of entitlement to shares of the committee. This ties back nicely to Chapter 3, as it can be interpreted as a partitioned approach: Each agent would represent a fraction of the voters proportional to its weight.

In the next chapter, we move on from the topic of cooperation to the topic of reconfiguration. We study how to reconfigure committees in the multi-winner approval voting setting.

CHAPTER 6

Multi-Winner Reconfiguration

6.1 Introduction

In the rapidly evolving landscape of technology, policy, and market dynamics, the ability to adapt and reconfigure existing solutions is essential. This necessity spans various domains, such as disaster management [IDH+11, JYW17]—where one needs to switch, step by step, from an existing power supply distribution to another one, while ensuring no blackouts occur in-between—and game and puzzle-solving [Nis18]—where one needs to transfer from one feasible state to another via a sequence of feasible moves. The crucial idea is to employ a systematic approach to transition from one solution to another with minimal disruption. Such a concept, known as *reconfiguration* [IDH+11], is omnipresent in areas ranging from everyday life to complex decision-making processes in computational social choice (COMSOC) such as switching catalogs in streaming services [PV24, Net24] or product displays in department stores.

This chapter delves into reconfiguration within COMSOC, using multi-winner elections as an example to illustrate its importance and applicability. While there have been dynamic approaches to multi-winner elections before, such as sequential elections, we are the first to study multi-winner elections through the lens of reconfiguration to the best of our knowledge. Our focus is on how to find such a reconfiguration through incremental yet meaningful changes, adhering to a set of criteria that quantify the "closeness" between successive solutions. As a reminder, in multi-winner elections, we are given a preference profile consisting of a finite set of alternatives and a finite set of voters, each with preferences over the alternatives. The objective is to select a committee (i.e., a subset of alternatives) of fixed size k, representing voters' preferences optimally (see Section 2.6). Briefly put, in *multi-winner reconfiguration* we need to decide whether there exists a transformation between two size-k (not necessarily optimal) committees via a sequence of intermediate size-k committees that are "almost" as good as the input committees and close to one another (see Section 6.2 for the formal definition). This can be understood

as dynamically adapting an existing solution, as described in Chapter 1. Consider the following possible real-world applications for multi-winner reconfiguration that require dynamic adaptation:

- Streaming service providers, such as Prime Video and Netflix, change their catalog frequently [PV24, Net24] to keep it engaging, while ensuring that popular or critically acclaimed content remains available. Here, customer *satisfaction* hinges on a balanced mix of fresh and familiar content. The multi-winner reconfiguration model can be used to optimize this balance, ensuring the catalog evolves in a way that minimizes disruption to viewer experience.
- Likewise, for physical or online stores, product displays are crucial in attracting and retaining customers. However, shelf or display space is generally limited. The challenge is to introduce new products or promotions without overshadowing popular or staple items. The reconfiguration model can guide the arrangement of products within a fixed capacity to maximize visibility of new items while ensuring customer satisfaction.
- An online gaming platform may need to shut down some old games to free up server space to support new games. However, abruptly removing a lot of games could alienate players, and thus the changes should be implemented incrementally in order to ensure player satisfaction throughout the whole exchange process.

In each of the above scenarios, the goal is to transition from one committee (i.e., a subset of products, movies, or games) to another through minimal, incremental changes, ensuring a balance between introducing new elements and maintaining overall user satisfaction. In other words, we need to ensure that the consecutive committees in the transition are close to each other, both in their content and optimality with respect to the initial committee. As an example, a small physical store with display space for exactly three goods may currently display apples, ketchup, and chocolate. The goal is to transition to displaying bananas, sour cream, and vanilla pudding. Rather than switching all goods at once—which might confuse or alienate customers—the store replaces apples with bananas in the first week, ketchup with sour cream in the second, and chocolate with vanilla pudding in the third. Each intermediate display differs from the previous one by only a single item and remains appealing to customers, forming what we will call a reconfiguration path.

Main contributions. We propose a model of multi-winner reconfiguration and study its computational complexity. Concretely, an instance of our model consists of a preference profile $\mathcal{P}$, a committee size k, two size-k committees W and W', and two numbers $\delta_c \geq 1$ and $\delta_s \geq 0$. The parameter δ_c upper-bounds the number of alternatives that can be exchanged in a single step, or equivalently, $2\,\delta_c$ bounds the symmetric difference between adjacent committees. The parameter δ_s bounds how much the score of an intermediate committee may drop below the score of the initial committee W. We

defer the formal definition to Section 6.2. We focus on four score-based multi-winner voting rules for approval preferences: Approval Voting (AV), Satisfaction Approval Voting (SAV), Chamberlin–Courant (CC), and Proportional Approval Voting (PAV). We find that a reconfiguration path always exists for the first two rules, while for the latter two, determining whether a reconfiguration path exists is PSPACE[1]-complete. We further investigate the parameterized complexity for CC and PAV by looking at several canonical parameters:

- the number m of alternatives,
- the number n of voters,
- the committee size k,
- the maximum number b of approved alternatives per voter,
- and the length ℓ of a shortest reconfiguration path (if it exists).

The parameters n, m, and k are natural input parameters that are already studied in the original multi-winner determination problem. The parameter b can be considered as a structural and distance to tractability parameter: For $\mathsf{b} = 1$, all studied voting rules are equivalent to AV, so by Theorem 6.7, the reconfiguration problem can be solved in polynomial time. Thus, it is natural to ask whether the problem is polynomial-time solvable for each constant b. Finally, by our reduction, the PSPACE-hardness arises because a shortest reconfiguration path could be of unbounded length. Note that for bounded length ℓ the problem is in NP, see Proposition 6.12. In real-world applications, one may need to reconfigure committees in a timely manner, so it is natural to assume that ℓ is small. We summarize our findings as follows (also see Table 6.1): For a definition of the relevant parameterized complexity classes, see Section 2.2.3:

1. For CC and PAV, the reconfiguration problem is FPT with respect to m. Under CC it is also FPT with respect to n.
2. While the reconfiguration problem is XP with respect to n (resp. k) for CC and PAV, it remains NP-hard even for constant value $\mathsf{b} + \ell$ for these rules. The XP result for k is essentially tight since it remains W[2]-hard under CC (resp. W[1]-hard under PAV).
3. Combining n with any other single parameter always yields an FPT algorithm for CC and PAV.

[1]See Definition 2.23 for a definition.

Table 6.1: (Parameterized) complexity results obtained in this chapter. We omit the results for AV and SAV as reconfiguration under both rules is polynomial-time solvable (see Theorem 6.7). Both PSPACE-hardness and W[1]-hardness hold even if each voter approves of at most two alternatives, i.e., $\mathsf{b} = 2$. Except for the NP-hardness for constant $\mathsf{b} + \ell$, all hardness results hold even if $(\delta_c, \delta_s) = (1, 0)$. Results in green are tractable, results in red are intractable ("-c" = complete, "-h" = hard); within a cell, two comma-separated entries give a lower and an upper bound, "?" marks an open case, and "–" a missing reference. References [T x] and [P x] point to Theorem and Proposition x of this chapter.

Parameter	CC		PAV	
In general	PSPACE-c	[T6.10]	PSPACE-c	[T6.10]
# alternatives m	FPT	[P6.13]	FPT	[P6.13]
# voters n	FPT	[T6.17]	?, XP	–, [P6.15]
committee size k	W[2]-h, XP	[T6.20], [P6.13]	W[1]-h, XP	[T6.10], [P6.13]
maximum ballot size b	PSPACE-h	[T6.10]	PSPACE-h	[T6.10]
path length ℓ	NP-h	[P6.12]	NP-h	[P6.12]
$\mathsf{b} + \ell$	NP-h	[P6.12]	NP-h	[P6.12]
$k + \ell$	W[2]-h, XP	[T6.20], [P6.13]	W[1]-h, XP	[T6.10], [P6.13]
$\mathsf{b} + k + \ell$	W[1]-h	[T6.10]	W[1]-h	[T6.10]
$n + \mathsf{b}$	FPT	[T6.17]	FPT	[P6.18]
$n + k$	FPT	[T6.17]	FPT	[P6.15]
$n + \ell$	FPT	[T6.17]	FPT	[T6.14]

Related work. Reconfiguration of puzzles and games has been of interest to mathematicians already since the 19th century: Johnson and Story [JS79] studied reachability in the setting of a sliding tile game. Since the early 2000s, reconfiguration has attracted increasing attention across many fields of theoretical computer science. In computational geometry, reconfiguration arises naturally for geometric structures such as planar graph drawings [AAB+17], polygonal linkages [CDR03], and polygon triangulations [AMP15].

In graph theory, reconfiguration has been studied for many classical vertex- and edge-subset problems—independent sets, vertex covers, dominating sets, and colorings [IDH+11]—as well as for path-based objects such as shortest paths [Bon13]. For graph problems, there are in general three types of reconfigurations between feasible solutions: token addition and removal (TAR; in each step, one can either remove or add a vertex), token jumping (TJ; in each step one can replace a vertex with another vertex), and token sliding (TS; in each step, one can replace a vertex with another adjacent vertex). Our multi-winner reconfiguration problem generalizes the TJ model by allowing the replacement of more than one alternative in each step. In constraint satisfaction, Gopalan et al. [GKMP09] obtained a classification of when the solution space of a Boolean formula

is connected under single-variable flips. Reconfiguration has even reached quantum complexity theory; see Gharibian and Sikora [GS18]. Usually, the reconfiguration variant of a problem—often PSPACE-complete—tends to be considerably harder than the underlying decision or optimization problem, which has motivated a rich body of work on tractable special cases and on parameterized algorithms. There are exceptions, however: Cereceda et al. [CvdHJ11] showed that 3-Coloring Reconfiguration is tractable, whereas 3-Coloring is NP-hard.

For a general introduction to reconfiguration, we refer to the surveys by van den Heuvel [vdH13] and Nishimura [Nis18]. We give a brief overview of other models that deal with multiple solutions and how they differ from reconfiguration:

- **Local search** [AEOP02] aims to improve a feasible solution through iterative moves to better solutions in a neighborhood defined by small modifications, until a local optimum is reached. This differs from reconfiguration in two important ways: First, reconfiguration fixes the endpoint to be reached, whereas local search just searches for a local optimum. Second, in local search, each move must lead to an improvement in solution quality, which is not the case in reconfiguration. Our model may even require an intermediate committee to be worse in order to reach the target committee.
- **Reoptimization** [SSTT18] studies a setting where a solution to an initial instance is given, and the task is to find a solution to a slightly modified version of the instance using the given solution. It therefore does not study sequences of solutions and also modifies the instance: two fundamental differences from reconfiguration.
- **Incremental problems** study settings where new parts of the input are revealed over time—often in an online setting—and a solution must be maintained. As an example, Bredereck et al. [BCK$^+$26] study an incremental problem in the setting of stable matching, where agents' preferences change over time, and new stable matchings should not differ too much. Incremental problems thus differ from reconfiguration in two ways: The instance evolves over time, and no specific target solution is fixed in advance.

Ito et al. [IKO$^+$14], Mouawad et al. [MNR$^+$17], and Bodlaender et al. [BGS26] studied the parameterized complexity of various reconfiguration problems, such as reconfiguring independent sets and dominating sets. They considered the length of the path as well as a bound on the size of the solutions as parameters. Notably, Bodlaender et al. claimed that Dominating Set Reconfiguration via Token Jumping is W[2]-hard with respect to the solution size and the reconfiguration path length by incorrectly referring to the TAR model. This is, in general, not correct as the two models are not equivalent. We correct this by providing a direct parameterized reduction from the Dominating Set problem (see Proposition 6.19).

Reconfiguration in computational social choice has recently gained traction, with Igarashi et al. [IKSY24] introducing the concept of reconfiguration in fair division. Ito et

al. [IIK+25] introduced a framework similar to reconfiguration for fair matchings. Dong et al. [DFPS26] studied reconfiguration for proportional committees. In their model, only committees that satisfy a fairness notion are admissible.

Multi-winner elections are a topic of great interest in computational social choice. In this chapter, we will only consider approval preferences and study multi-winner elections from a dynamic perspective. In Chapter 5, we also study multi-winner elections, but from a game-theoretic perspective.

Reconfiguration is related to the model of "one problem, successive solutions" by Boehmer and Niedermeier [BN21], where the goal is to select a sequence of desirable solutions. Bredereck et al. [BKN20] introduced and studied this model for a multi-winner election where each alternative in the committee must remain in office for a consecutive number of rounds. Our reconfiguration problem differs from theirs in two ways: First, we aim to obtain a sequence of committees starting from an initial committee and ending at a target committee, such that the intermediate committees are "close" to each other. Second, we do not require that an alternative appears a limited number of times.

6.2 Preliminaries

Reconfiguration of committees. Let $\mathcal{P}$ be a profile and λ a voting rule. Given a number d, we say that two committees W and W' are *d-adjacent* if $|W \setminus W'| \leq d$ and $|W' \setminus W| \leq d$. For two numbers $\delta_c \geq 1$ and $\delta_s \geq 0$, we call a sequence of committees $W_0, \ldots, W_t$ a *(δ_c, δ_s)-reconfiguration path* (*reconfiguration path* for short) for $(\mathcal{P}, \lambda)$ if for all $i \in [t]$ the following holds:

(C1) $|W_{i-1}| = |W_i|$;

(C2) W_{i-1} and W_i are δ_c-adjacent;

(C3) The score difference of W_0 and W_i is bounded by δ_s, i.e., $\mathrm{sc}_\lambda(W_0) - \mathrm{sc}_\lambda(W_i) \leq \delta_s$.

Remark 6.1. *Note that the reconfiguration path requires closeness (C2–C3) between committees of the same size (C1). This is essential, as closeness captures the incremental yet necessary changes across various applications, such as evolving content in streaming services or adjusting product line-ups in retail stores, while fixing the committee size (C1) mirrors practical limitations, such as preset numbers of movie slots in streaming catalogs or specific shelf capacities in retail stores.*

By convention, the *length* of a reconfiguration path is defined as the number of committees on the path minus one.

Example 6.2. *Let $\mathcal{C} = [4]$ and $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ with approval preferences $\mathsf{A}_1 = \{1, 2\}, \mathsf{A}_2 = \{2, 3\}, \mathsf{A}_3 = \{3, 4\}$ and $\mathsf{A}_4 = \{1, 4\}$. Let $k = 2$. Under both CC and PAV, we have two size-2 committees with maximum score 4: $W = \{1, 3\}$ and $W' = \{2, 4\}$.*

For $\delta_c = \delta_s = 1$ and under CC, there is a (δ_c, δ_s)-reconfiguration path between W and W' of length two, e.g., $(W, \{1,2\}, W')$. The intermediate committee $\{1,2\}$ has CC-score 3, only one below the maximum, so it satisfies the score difference bound $\delta_s = 1$ of condition (C3); each step also exchanges a single alternative, satisfying (C2) with $\delta_c = 1$. As all committees have the same size, (C1) is satisfied as well. The same path is also a $(1, \frac{1}{2})$-reconfiguration path under PAV since $sc_{PAV}(\{1,2\}) = \frac{7}{2}$.

Problem definition. Now, we define our reconfiguration problem.

λ-Multi-Winner Reconfiguration (λ-*MR*)
Input: A profile $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$, a committee size k, two numbers $\delta_c \geq 1$ and $\delta_s \geq 0$, and two size-k committees $W \neq W'$ with $\mathrm{sc}_\lambda(W) - \mathrm{sc}_\lambda(W') \leq \delta_s$.
Question: Does there exist a sequence $(W_0, \ldots, W_t)$ of committees which is a (δ_c, δ_s)-reconfiguration path for $(\mathcal{P}, \lambda)$ such that $W_0 = W$ and $W_t = W'$?

Remark 6.3. *Note that we require that the input satisfies $\delta_c \geq 1$, $W \neq W'$, and $sc_\lambda(W) - sc_\lambda(W') \leq \delta_s$, as otherwise either there is always a reconfiguration path (when $W = W'$) or none (when $\delta_c < 1$ and $W \neq W'$, or $sc_\lambda(W) - sc_\lambda(W') > \delta_s$). Moreover, we intentionally do not restrict the committees to be optimal (i.e., winning). This choice allows for a more realistic representation of scenarios where optimal solutions are not immediate or feasible, thereby enhancing the practical relevance of our study.*

We write λ-MR^0 to refer to the restricted variant where the committees on the reconfiguration path must be optimal (i.e., W and W' are winning and $\delta_s = 0$) and we can only exchange one alternative in each step (i.e., $\delta_c = 1$). We will see that all intractability results except Proposition 6.12 already hold for the restricted case, while all algorithmic results hold for the general case.

Parameters. In this chapter, we look at the problems through the lens of parameterized complexity using the following parameters:

- the number m of alternatives,
- the number n of voters,
- the committee size k,
- the maximum number b of approved alternatives per voter,
- and the length ℓ of a shortest reconfiguration path (if it exists).

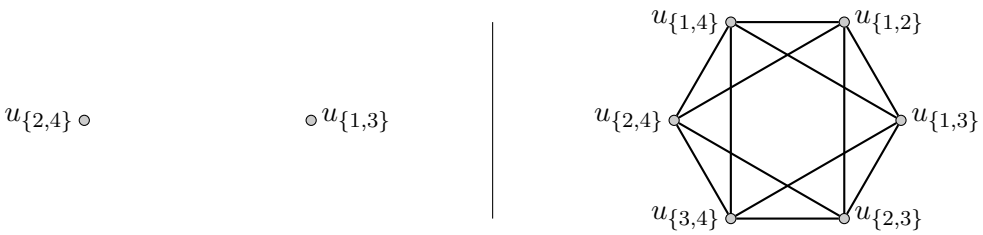


Figure 6.1: Reconfiguration graphs for Example 6.5 under CC and PAV. Left: $\delta_s = 0$ under both CC and PAV. Right: $\delta_s = 1$ under CC or $\delta_s = \frac{1}{2}$ under PAV.

Reconfiguration graphs. Let $I = (\mathcal{P}, k, \delta_c, \delta_s, W, W')$ be an instance of λ-MR with $\mathcal{P} = (\mathcal{C} = [m], \mathcal{V}, \mathcal{R})$ being a profile and λ a score-based voting rule. We construct a *reconfiguration graph* $G^{\mathsf{R}} = (V^{\mathsf{R}}, E^{\mathsf{R}})$ for I as follows. Let $V^{\mathsf{R}} = \{u_{\mathcal{C}'} \mid \mathcal{C}' \subseteq \mathcal{C} \text{ s.t. } |\mathcal{C}'| = k \text{ and } \mathrm{sc}_\lambda(\mathcal{P}, W) - \mathrm{sc}_\lambda(\mathcal{P}, \mathcal{C}') \leq \delta_s\}$ be the vertex set of G^{R} such that each vertex corresponds to a size-k committee whose score is at most δ_s lower than the score of W. For each pair u_{W_1}, u_{W_2} of vertices, we create an edge $\{u_{W_1}, u_{W_2}\}$ if the corresponding two committees W_1 and W_2 are δ_c-adjacent.

Remark 6.4. *Note that $|V^{\mathsf{R}}| \leq \binom{m}{k}$ since only committees of size k are relevant. We can compute V^{R} in time $\binom{m}{k} \cdot \mathsf{T}(\mathcal{P}, \lambda)$, where $\mathsf{T}(\mathcal{P}, \lambda)$ denotes the time to compute the score of a committee under rule λ. The number of edges of G^{R} is bounded above by $\binom{m}{k}^2$, so the reconfiguration graph can be constructed in time $O(\binom{m}{k}^2 + \binom{m}{k} \cdot \mathsf{T}(\mathcal{P}, \lambda))$. By construction, if $\delta_s = 0$, and both W and W' are winning committees, then for each pair of connected vertices u_{W_1} and u_{W_2}, there exists a one-to-one correspondence between all (u_{W_1}, u_{W_2})-paths in G^{R} and all (δ_c, δ_s)-reconfiguration paths for W_1 and W_2.*

Example 6.5. *Let us consider the instance given in Example 6.2. If $\delta_s = 0$, then the vertex set of the reconfiguration graph G^{R} is $V^{\mathsf{R}} = \{u_{\{1,3\}}, u_{\{2,4\}}\}$ since $\{1,3\}$ and $\{2,4\}$ are the only two winning committees. Because $|\{1,3\}\Delta\{2,4\}| = 4$, if $\delta_c = 1$, there is no path between u_W and $u_{W'}$ and hence no reconfiguration path from W to W' exists. See the LHS of Figure 6.1 for an illustration.*

If $\delta_s = 1$ (under CC) or $\delta_s = \frac{1}{2}$ (under PAV), then V^{R} contains vertices for all size-2 committees of $\mathcal{C}$. There are four possible length-2 paths between $u_{\{1,3\}}$ and $u_{\{2,4\}}$, for example $(u_{\{1,3\}}, u_{\{1,2\}}, u_{\{2,4\}})$. See the RHS of Figure 6.1 for an illustration. Note that the length of a shortest path between any two committees W, W' is at least $\left\lceil \frac{|W \Delta W'|}{2\,\delta_c} \right\rceil$; any edge on the reconfiguration graph will remove (resp. add) at most δ_c alternatives from $W \setminus W'$ (resp. $W' \setminus W$).

6.3 Complexity Results

In this section, we consider the computational and parameterized complexity of λ-MR.

Computational complexity. It is straightforward that λ-MR is in PSPACE due to an observation by Ito et al. [IDH+11].

Proposition 6.6. *For each λ of the four considered voting rules, λ-MR is in* PSPACE.

Proof. It suffices to show that λ-MR is contained in NPSPACE since PSPACE = NPSPACE [Sav70]. Moreover, as observed by Ito et al. [IDH+11], to show that the reconfiguration version of a problem is in NPSPACE, one only needs to show that the underlying problem is in NP. In our case, we need to show that the following two problems can be verified in polynomial time:

1. Does a given committee satisfy a given score-bound?
2. Are two given committees δ_c-adjacent?

For the sake of completeness, we provide a detailed proof for λ-MR.

Starting with W, we can iteratively non-deterministically choose a size-k committee W'' that is δ_c-adjacent to the current committee and has $\mathrm{sc}_\lambda(\mathcal{P}, W) - \mathrm{sc}_\lambda(\mathcal{P}, W'') \leq \delta_s$. This is possible, as we can verify in polynomial time whether the committee W'' is δ_c-adjacent and whether W'' satisfies the score-bound. The algorithm returns yes if we reach the target committee W'. Since there are $\binom{m}{k} \leq m^k$ possible committees, the length of a shortest reconfiguration path is bounded by m^k. We can keep a counter that tracks how many committees we have traversed—in a binary encoding, this uses polynomial space—and the algorithm returns no if we have not reached W' when the counter reaches m^k. The problem is thus in NPSPACE. □

The following result implies that AV- and SAV-MR are easy. This is because both rules are based on *additive* scoring functions, so we can greedily exchange alternatives from the symmetric difference between the initial and target committees. Informally, swapping the lowest-scoring alternative in $W \setminus W'$ out for the highest-scoring alternative in $W' \setminus W$ in each step never decreases the score of the resulting committee beyond the minimum of $\mathrm{sc}_\lambda(W)$ and $\mathrm{sc}_\lambda(W')$.

Theorem 6.7. *Every instance of AV-MR (resp. SAV-MR) admits a reconfiguration path. Moreover, a shortest one can be computed in polynomial time and has length $\lceil \frac{|W \setminus W'|}{\delta_c} \rceil = \lceil \frac{|W \Delta W'|}{2\,\delta_c} \rceil$.*

Proof. Let $\lambda \in \{\mathrm{AV}, \mathrm{SAV}\}$ and $I = (\mathcal{P}, k, \delta_c, \delta_s, W, W')$ denote an instance of λ-MR. As already discussed, we will greedily exchange alternatives from $W \Delta W'$. The correctness is based on the following claims:

Claim 6.8. *Let W_1 and W_2 be two size-k committees, and $a = \arg\max_{c \in W_2 \setminus W_1} \mathrm{sc}_\lambda(c)$ and $b = \arg\min_{c \in W_1 \setminus W_2} \mathrm{sc}_\lambda(c)$. Then, $\mathrm{sc}_\lambda((W_1 \cup \{a\}) \setminus \{b\}) \geq \min\{\mathrm{sc}_\lambda(W_1), \mathrm{sc}_\lambda(W_2)\}$.*

Proof. We distinguish between two cases: $\mathrm{sc}_\lambda(a) \geq \mathrm{sc}_\lambda(b)$, and $\mathrm{sc}_\lambda(a) < \mathrm{sc}_\lambda(b)$.

If $\mathrm{sc}_\lambda(a) \geq \mathrm{sc}_\lambda(b)$, due to the additivity of the voting rule, the statement follows: $\mathrm{sc}_\lambda(W_1 \cup \{a\} \setminus \{b\}) = \mathrm{sc}_\lambda(W_1) + \mathrm{sc}_\lambda(a) - \mathrm{sc}_\lambda(b) \geq \mathrm{sc}_\lambda(W_1) \geq \min\{\mathrm{sc}_\lambda(W_1), \mathrm{sc}_\lambda(W_2)\}$.

If $\mathrm{sc}_\lambda(a) < \mathrm{sc}_\lambda(b)$, by the choice of a and b, we infer that $\mathrm{sc}_\lambda(c) < \mathrm{sc}_\lambda(d)$ for all $c \in W_2 \setminus W_1$ and all $d \in W_1 \setminus W_2$. We now enumerate the elements in $W_2 \setminus W_1$ in the order $a_1, \ldots, a_t$ so that $\mathrm{sc}_\lambda(a_i) \geq \mathrm{sc}_\lambda(a_{i+1})$ holds for all $i \in [t-1]$. Similarly, we enumerate $W_1 \setminus W_2$ in the order $b_1, \ldots, b_t$ so that $\mathrm{sc}_\lambda(b_i) \geq \mathrm{sc}_\lambda(b_{i+1})$ holds for all $i \in [t-1]$. Note that in these orderings we have $a = a_1$ and $b = b_t$. Using this enumeration, we can now show the statement:

$$
\begin{aligned}
& \mathrm{sc}_\lambda(W_1 \cup \{a\} \setminus \{b\}) = \mathrm{sc}_\lambda(W_1) + \mathrm{sc}_\lambda(a) - \mathrm{sc}_\lambda(b) \\
= \;& \mathrm{sc}_\lambda(W_1 \cap W_2) + \textstyle\sum_{i \in [t]} \mathrm{sc}_\lambda(b_i) + \mathrm{sc}_\lambda(a) - \mathrm{sc}_\lambda(b) \\
= \;& \mathrm{sc}_\lambda(W_1 \cap W_2) + \textstyle\sum_{i \in [t-1]} \mathrm{sc}_\lambda(b_i) + \mathrm{sc}_\lambda(a) \\
\geq \;& \mathrm{sc}_\lambda(W_1 \cap W_2) + \textstyle\sum_{i \in [t-1]} \mathrm{sc}_\lambda(a_i) + \mathrm{sc}_\lambda(a) \\
\geq \;& \mathrm{sc}_\lambda(W_1 \cap W_2) + \textstyle\sum_{i \in [t]} \mathrm{sc}_\lambda(a_i) = \mathrm{sc}_\lambda(W_2) \qquad \text{(of Claim 6.8)} \diamond
\end{aligned}
$$

By Claim 6.8, we can iteratively replace the lowest scoring alternative in $W \setminus W'$ with the highest scoring alternative in $W' \setminus W$, while ensuring the overall score is at least $\min\{\mathrm{sc}_\lambda(W), \mathrm{sc}_\lambda(W')\}$. Since $\mathrm{sc}_\lambda(W) \leq \mathrm{sc}_\lambda(W') + \delta_\mathrm{s}$ by definition, we obtain a valid reconfiguration path in polynomial time. For $\delta_\mathrm{c} = 1$, such a path has the shortest possible length $|W \setminus W'|$. For $\delta_\mathrm{c} > 1$, we can take the same path, but skip $\delta_\mathrm{c} - 1$ many committees in each step. This leads to a path of length $\lceil \frac{|W \setminus W'|}{\delta_\mathrm{c}} \rceil$. Note that a shorter path cannot exist. □

Example 6.9. *We illustrate the greedy approach behind the proof of Theorem 6.7 on the following example under AV: Let* $\mathcal{C} = [6]$ *and* $\mathcal{V} = \{v_1, \ldots, v_{14}\}$ *with approval preferences* $\mathsf{A}_1 = \cdots = \mathsf{A}_5 = \{1,2,3,4\}$, $\mathsf{A}_6 = \mathsf{A}_7 = \mathsf{A}_8 = \{3,4,6\}$, $\mathsf{A}_9 = \mathsf{A}_{10} = \{2,5\}$, *and* $\mathsf{A}_{11} = \cdots = \mathsf{A}_{14} = \{1,5,6\}$. *We have the following scores for each of the alternatives:* $\mathrm{sc}(1) = 9$, $\mathrm{sc}(2) = 7$, $\mathrm{sc}(3) = \mathrm{sc}(4) = 8$, $\mathrm{sc}(5) = 6$, *and* $\mathrm{sc}(6) = 7$. *Let* $k = 3$, $\delta_\mathrm{c} = 1$, $\delta_\mathrm{s} = 5$, $W = \{1,3,4\}$, *and* $W' = \{2,5,6\}$. *Note that* W *is a winning committee with* $\mathrm{sc}(W) = 25$ *and* $\mathrm{sc}(W') = 20$.

We now proceed with generating the reconfiguration path. We start with $W_0 = W$. *Alternative* 6 *(or* 2*) is the highest scoring alternative in* $W' \setminus W_0$, *while alternative* 4 *(or* 3*) is the lowest scoring alternative in* $W_0 \setminus W'$. *Therefore, we generate* $W_1 = \{1,3,6\}$ *by exchanging* 4 *with* 6. *Using the same procedure, we compute* $W_2 = \{1,2,6\}$, *and finally* $W_3 = \{2,5,6\} = W'$. *It is straightforward to check that* $(W_0 = W, W_1, W_2, W_3 = W')$ *is a* $(1,5)$*-reconfiguration path.*

In contrast to AV and SAV, the next theorem shows that CC-MR and PAV-MR are computationally difficult. The idea is to reduce from the PSPACE-complete and W[1]-hard problem Independent Set Reconfiguration via Token Jumping (ISR-TJ) [IKO+14], by adapting from the reduction of Aziz et al. [AGG+15] for CC and PAV.

Ito et al. [IKO+14] show that Independent Set Reconfiguration via Token Jumping[2] is W[1]-hard with respect to the number of tokens (i.e., the size of independent sets on the path). As a reminder, an independent set of a graph is a subset of vertices such that no two of them are adjacent (cf. Definition 2.5).

Theorem 6.10. *For all $\lambda \in \{CC, PAV\}$, λ-MR^0 (and hence λ-MR) is* PSPACE*-complete. Moreover, they are* PSPACE*-hard and W[1]-hard with respect to $k + \ell$, even if $(\delta_c, \delta_s, \mathsf{b}) = (1, 0, 2)$.*

Proof. Since PSPACE-membership follows from Proposition 6.6, we only need to focus on the hardness results. We show W[1]-hardness and PSPACE-hardness by reducing from ISR-TJ (see Section 2.7). Ito et al. [IKO+14] show that ISR-TJ is both PSPACE-complete and W[1]-hard with respect to h, the number of tokens. We observe that the reduction is simultaneously a polynomial-time and W[1]-hardness reduction with respect to $h + \hat{\ell}$ because a shortest reconfiguration path has length $2h - 1$ in their construction (see Theorem 2 in [IKO+14]). Hence, if we can provide a polynomial-time reduction for CC-MR^0 and PAV-MR^0 such that the committee size and the path length are h and $\hat{\ell}$, respectively, then we obtain W[1]-hardness in the statement.

Let (G, h, I_s, I_t) be an instance of ISR-TJ. We create an instance of CC-MR^0 and PAV-MR^0 with committee size $k = h$. For each vertex $v_i \in V(G)$, let $\deg(v_i)$ denote the degree of vertex v_i in G. Further, let $\deg(G) \coloneqq \max_{v_i \in V(G)} \deg(v_i)$ denote the maximum degree of graph G.

- For each vertex $v_i \in V(G)$, we create a *vertex-alternative* c_i and $(\deg(G) - \deg(v_i))$-many private dummy voters $\{d_i^1, \dots, d_i^{\deg(G)-\deg(v_i)}\}$ that only approve c_i.
- For each edge $\{v_i, v_j\} \in E(G)$, we create one *edge-voter* $u_{\{i,j\}}$ that approves only c_i and c_j.

Let $\mathcal{P}$ denote the created profile with $\mathcal{C} = \{c_i \mid v_i \in V(G)\}$ and $\mathcal{V} = \{d_i^z \mid v_i \in V(G), z \in [\deg(G) - \deg(v_i)]\} \cup \{u_{i,j} \mid \{v_i, v_j\} \in E(G)\}$. The total number of voters created is $|\mathcal{V}| = \deg(G) \cdot |V(G)| - |E(G)|$. To complete the construction, let the initial and target committees be $W = \{c_i \mid v_i \in I_s\}$ and $W' = \{c_i \mid v_i \in I_t\}$, and the committee size $k = h$.

For the sake of brevity, define a *bijection* $f \colon 2^{V(G)} \to 2^{\mathcal{C}}$ with $f(V') = \{c_i \mid v_i \in V'\}$ between the vertex subsets and the committees. We present a one-to-one correspondence between the size-h independent sets in G and size-h committees with score of at least $\deg(G) \cdot k$ in the created PAV-MR^0- and CC-MR^0-instance.

Claim 6.11. *V' is a size-h independent set if and only if $f(V')$ is a size-h committee of score at least $\deg(G) \cdot k$ under CC as well as PAV.*

[2] See Section 2.7 for a definition.

Proof. The "only if" direction is straightforward: For each size-h independent set V', we have that the alternatives in $f(V')$ are approved by $\deg(G) \cdot k$ voters. Among them $\deg(G) \cdot k - \sum_{c_i \in f(V')} \deg(v_i)$ voters are private voters and $\sum_{c_i \in f(V')} \deg(v_i)$ voters are the edge-voters of the edges incident to V'. Each of these $\sum_{c_i \in f(V')} \deg(v_i)$ edge-voters approves exactly one alternative in $f(V')$ since V' is an independent set. Thus $\mathrm{sc}_{\mathrm{PAV}}(f(V')) = \mathrm{sc}_{\mathrm{CC}}(f(V')) = \deg(G) \cdot k$.

For the "if" direction, let W'' denote a size-h committee with score $\geq \deg(G) \cdot k$. We claim that $f^{-1}(W'')$ is a size-h independent set. Clearly $|f^{-1}(W'')| = h$, so we only need to show that no edge is incident to more than one vertex in $f^{-1}(W'')$. Let $E_1 = \{\{v_i, v_j\} \in E(G) \colon |\{v_i, v_j\} \cap f^{-1}(W'')| = 1\}$ and $E_2 = \{\{v_i, v_j\} \in E(G) \colon |\{v_i, v_j\} \cap f^{-1}(W'')| = 2\}$ denote the set of edges that are incident to exactly one vertex in $f^{-1}(W'')$ and the set of edges that are incident to two vertices in $f^{-1}(W'')$, respectively. Then, by the handshaking lemma, we have that

$$|E_1| + 2|E_2| = \sum_{c_i \in W''} \deg(v_i). \tag{6.1}$$

For CC, we have that $\deg(G) \cdot k = \mathrm{sc}_{\mathrm{CC}}(W'') = \sum_{c_i \in W''} \big(\deg(G) - \deg(v_i)\big) + |E_1| + |E_2|$, which implies that $|E_1| + |E_2| = \sum_{c_i \in W''} \deg(v_i)$ since $k = h$. Together with (6.1), we obtain that $|E_2| = 0$, meaning that no edge is incident to more than one vertex in $f^{-1}(W'')$, as desired.

As for PAV, since $\deg(G) \cdot k = \mathrm{sc}_{\mathrm{PAV}}(W'') = \sum_{c_i \in W''} \big(\deg(G) - \deg(v_i)\big) + |E_1| + (1 + 1/2)|E_2|$, we infer that $|E_1| + (1 + 1/2)|E_2| = \sum_{c_i \in W''} \deg(v_i)$. Similarly to the previous case, by (6.1), we obtain that $|E_2| = 0$, meaning that no edge is incident to more than one vertex in $f^{-1}(W'')$, as desired. (of Claim 6.11) $\diamond$

By Claim 6.11, it is straightforward to verify that a sequence $(I_s, \ldots, I_t)$ of vertex subsets is a size-h independent set reconfiguration path if and only if $(W = f(I_s), \ldots, W' = f(I_t))$ is a $(1, 0)$-reconfiguration path for CC-MR and PAV-MR. This concludes the proof for the PSPACE-hardness since the reduction is a polynomial-time reduction.

Note that in our construction, each voter approves of at most 2 alternatives, as each edge-voter approves only the two vertex-alternatives corresponding to its endpoints, and each private dummy voter only approves one vertex-alternative. This means the PSPACE-hardness remains even if the ballot size is bounded by 2.

It remains to show the W[1]-hardness with respect to the committee size k and the path length ℓ. As the committee size k is equal to the size h of the independent set, each committee on the reconfiguration path directly corresponds to an independent set on a reconfiguration path of the size-h independent set, and the shortest path is of length at most $2h - 1$ [IKO+14], it follows that both PAV-MR^0 and CC-MR^0 are W[1]-hard with respect to the path length ℓ and the committee size k. This concludes the proof. $\square$

We observe that the reduction of Ito et al. [IKO+14], used to show that ISR-TJ is W[1]-hard with respect to the number of tokens, can also be used to show that a generalized

variant of ISR-TJ—where in each step more than one vertex can be exchanged—is NP-hard for constant path length. Hence, we obtain the following hardness for our problem as well.

Proposition 6.12. *For all $\lambda \in \{CC, PAV\}$, λ-MR remains NP-hard even if $\mathsf{b} = 2$ and $\ell = 3$, and for constant ℓ, it is in NP.*

Proof. As already mentioned, the reduction of Ito et al. [IKO+14] can be used to show NP-hardness for the case where we allow for more than one token to be moved at once. In other words, it is NP-hard to decide whether one can reconfigure from one independent set I_s of size h to another independent set I_t of size h via a constant number of intermediate steps. To this end, we show that it is NP-hard to decide whether one can reconfigure from an independent set I_s of size $2h$ to an independent set I_t of size $2h$ via a reconfiguration path with $\ell \leq 3$, i.e., with at most two intermediate independent sets, where up to h tokens may be moved simultaneously. We then obtain the result by using the same reduction as for Theorem 6.10.

We reduce from Independent Set[3]. Let $(G = (V, E), h)$ be an instance of Independent Set. We construct an instance of ISR-TJ on a graph H consisting of two disjoint components G_1 and G_2, where $G_1 = G$ and $G_2 = K_{2h,2h}$ is the complete bipartite graph with sides A and B, each of size $2h$. We set the initial independent set to $I_s = A$ and the target independent set to $I_t = B$, and allow up to h tokens to be moved simultaneously, i.e., $\delta_c = h$.

If G admits an independent set U of size h, then the following is a reconfiguration path of length three: Move h tokens from A to the vertices of U in G_1; move the remaining h tokens from A to B; finally, move the h tokens from G_1 to B.

Conversely, suppose no independent set of size h exists in G. Since G_1 and G_2 are disjoint and every vertex of A is adjacent to every vertex of B in $K_{2h,2h}$, no independent set of H can contain vertices from both A and B simultaneously. Therefore, to transition from $I_s = A$ to $I_t = B$, there must exist some intermediate independent set containing no tokens on A. At that point, the h tokens not yet on B must reside in G_1 and form an independent set of size h in G, which is a contradiction. Hence, no reconfiguration path exists.

As for constant path length ℓ, we can guess a sequence of constantly many committees which in total has polynomial size and check in polynomial time whether it is a valid reconfiguration path. This proves containment in NP. □

Parameterized complexity. Next, we focus on the parameterized complexity for CC and PAV with respect to a few selected parameters. The most intuitive ones are the number m of alternatives, the number n of voters, and the committee size k.

[3]See Section 2.7 for a definition and additional information.

Proposition 6.13. *For all score-based multi-winner voting rules λ such that the score of a committee can be computed in polynomial time, λ-MR is* FPT *with respect to m and* XP *with respect to k, and can be solved in $m^{2k} \cdot (n+m)^{O(1)}$ time. Furthermore, a shortest path can be computed in the same time.*

Proof. Let $I = (\mathcal{P}, k, \delta_c, \delta_s, W, W')$ be an instance of λ-MR. We operate on the reconfiguration graph G^{R} of I. As noted in Remark 6.4, the number of edges in G^{R} is bounded above by $\binom{m}{k}^2 \leq m^{2k} \leq m^{2m}$, and hence G^{R} can be constructed in $\binom{m}{k}^2 \cdot (n+m)^{O(1)}$ time. Since determining a shortest path between two vertices can be done in time linear in the number of edges of the graph, we can determine a shortest-length $(u_W, u_{W'})$-path in G^{R} in $m^{2k} \cdot (n+m)^{O(1)}$ time, which is FPT with respect to m and XP with respect to k, as desired. □

Yang and Wang [YW23] show that PAV-Multi-Winner Score can be solved in FPT-time with respect to n via a specific integer linear programming (ILP) formulation called Integer Programming With Simple Piecewise Linear Transformation (IPWSPLT) that can encode convex constraints.[4] It is not clear how to extend their approach to the reconfiguration variant because the length ℓ of a shortest reconfiguration path may be unbounded. However, we can extend their approach to obtain an FPT result with respect to $n + \ell$.

Theorem 6.14. *PAV-MR is* FPT *with respect to $n + \ell$.*

Proof. As mentioned above, we extend the IPWSPLT by Yang and Wang [YW23]. Briefly put, IPWSPLT is an ILP extension that allows for piecewise concave or convex functions to be used [BFN+20]. We construct ℓ copies of the IPWSPLT formulation and combine them with additional constraints to model the exchanges. Informally, the ℓ copies each correspond to one step in the reconfiguration path. By adding constraints that bound the number of exchanged alternatives and ensuring that the initial and target committees are represented by the first and last committees of the path, respectively, we obtain our ILP formulation. Similarly to their approach, we utilize the fact that alternatives of the same type are interchangeable when considering the score of a committee. We then split the alternatives of each type[5] into 4 *groups*: alternatives that are in $W \cap W'$, $W \setminus W'$, $W' \setminus W$, and $\mathcal{C} \setminus (W \cup W')$. We add variables that keep track of the number of alternatives of each group from each type. By keeping track of these numbers along the steps of the reconfiguration path, we ensure that the symmetric difference between two intermediate committees is at most $2\,\delta_c$ and that the final committee in the reconfiguration path is the target committee. We ensure that the total number of variables and constraints is in

[4] ILP with convex constraints was shown to be FPT with respect to n before the introduction of IPWSPLT by Grötschel et al. [GLS88], but we use IPWSPLT to keep the terminology consistent with Yang and Wang.

[5] Reminder: A type of alternative is defined by the set of voters that approve them. I.e., two alternatives are of the same type if they are approved by the exact same set of voters.

$O(4^n \cdot \ell)$. This yields an FPT algorithm with respect to $n + \ell$ since an IPWSPLT instance can be solved in time $O(\hat{n}^{2.5\hat{n}+o(\hat{n})} \cdot (|I| + p_{\max}))$ [BFN+20], where $\hat{n}$ is the number of integer variables, $|I|$ is the size of the encoding of the instance, and $p_{\max}$ is the maximum number of pieces of a piecewise-linear convex/concave function. Since in our IPWSPLT formulation the number $\hat{n}$ of variables and the size of the encoding $|I|$ are in $O(4^n \cdot \ell)$, the FPT result follows.

We first introduce some notations and concepts that are useful for the IPWSPLT formulation. For each type t of alternative, let C_t denote the set consisting of all alternatives with type t. Let $\mathcal{T}$ denote the set of types of alternatives that exist in a given preference profile $\mathcal{P}$. Further, let $A(t)$ denote the set of voters that approve of alternatives of type t. We say that two committees C_1 and C_2 are in the same *equivalence class* (*class* for short) if for each type t, the number of alternatives with type t in C_1 is the same as that in C_2. We partition the set C_t of alternatives of type t into four subsets as follows: $X_t^1 = \mathsf{C}_t \cap (W \setminus W')$, $X_t^2 = \mathsf{C}_t \cap (W \cap W')$, $X_t^3 = \mathsf{C}_t \cap (W' \setminus W)$, and $X_t^4 = \mathsf{C}_t \cap (\mathcal{C} \setminus (W' \cup W))$. In other words, X_t^1, X_t^2, X_t^3, and X_t^4 denote the set consisting of all alternatives of type t that are from the set $W \setminus W'$, the intersection $W \cap W'$, the other set $W' \setminus W$, and not in $W \cup W'$, respectively.

We now describe the variables and constraints used in the IPWSPLT. We use IPWSPLT rather than a regular ILP because it is not clear how to compute the scores under PAV with just ILP.

Variables. The following variables will be used:

- An integer variable $x_{t,j}^1$ for each $t \in \mathcal{T}$ and each $j \in \{0, \dots, \ell\}$. This variable stores the number of alternatives in X_t^1 in the j-th committee of the reconfiguration path. There are at most $2^n \cdot \ell$ variables of this kind.
- An integer variable $x_{t,j}^2$ for each $t \in \mathcal{T}$ and each $j \in \{0, \dots, \ell\}$. This variable stores the number of alternatives in X_t^2 in the j-th committee of the reconfiguration path. There are at most $2^n \cdot \ell$ variables of this kind.
- An integer variable $x_{t,j}^3$ for each $t \in \mathcal{T}$ and each $j \in \{0, \dots, \ell\}$. This variable stores the number of alternatives in X_t^3 in the j-th committee of the reconfiguration path. There are at most $2^n \cdot \ell$ variables of this kind.
- An integer variable $x_{t,j}^4$ for each $t \in \mathcal{T}$ and each $j \in \{0, \dots, \ell\}$. This variable stores the number of alternatives in X_t^4 in the j-th committee of the reconfiguration path. There are at most $2^n \cdot \ell$ variables of this kind.
- An integer variable $y_{(t,j-1),(t',j)}^{z,z'}$ for each $z, z' \in [4]$, $j \in [\ell]$ and $t, t' \in \mathcal{T}$. This variable stores the number of alternatives that were part of the group represented by $x_{t,j-1}^z$ that were replaced by alternatives in the group $x_{t',j}^{z'}$ in the step from W_{j-1} to W_j. There are at most $16 \cdot 4^n \cdot \ell$ variables of this kind.

- An integer variable $x_{v,j}$ for each voter $v \in V$ and each step $j \in \{0, \dots, \ell\}$. This variable stores how many alternatives voter v approves of in committee W_j. There are $n \cdot \ell$ variables of this kind.

We use the same piecewise linear concave function $f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as Yang and Wang. We set $f(0) = 0$ and for every positive integer x we set $f(x) = \sum_{i=1}^{x} \frac{1}{i}$. Then for each real positive non-integer x we choose $y \in \mathbb{N}$ such that $y = \lfloor x \rfloor$ and set $f(x) = f(y) + (x - y) \cdot (f(y+1) - f(y))$.

Constraints. We add the following constraints:

1. For each $z \in [4]$ and $t \in \mathcal{T}$, we fix $x^z_{t,0}$ and $x^z_{t,\ell}$ to be the correct values for W and W', i.e., $x^1_{t,0} = |X^1_t|$, $x^2_{t,0} = x^2_{t,\ell} = |X^2_t|$, $x^3_{t,0} = x^4_{t,0} = x^1_{t,\ell} = x^4_{t,\ell} = 0$, and $x^3_{t,\ell} = |X^3_t|$. This adds $8 \cdot |\mathcal{T}|$ equality constraints.

2. For each $j \in [\ell]$, we add constraint $\sum_{\substack{z,z' \in [4] \\ t,t' \in \mathcal{T}}} y^{z,z'}_{(t,j-1),(t',j)} \leq \delta_{\mathrm{c}}$. These constraints ensure that the symmetric difference between consecutive committees along the reconfiguration path remains within the bound. This adds ℓ constraints.

3. For each $(j, t, z) \in \{0, \dots, \ell\} \times \mathcal{T} \times [4]$, we add constraint $x^z_{t,j} \leq |X^z_t|$. These constraints ensure that the class of committee used in an intermediate step exists, i.e., we do not use more alternatives of a type in a committee than possible. This adds $4 \cdot 2^n \cdot (\ell + 1)$ constraints.

4. For each $(j, t, z) \in [\ell] \times \mathcal{T} \times [4]$, we add a constraint $x^z_{t,j-1} \geq \sum_{\substack{t' \in \mathcal{T} \\ z' \in [4]}} y^{z,z'}_{(t,j-1),(t',j)}$. These constraints are added so that it is not possible to switch out alternatives that are not in the committee. This adds $4 \cdot 2^n \cdot \ell$ constraints.

5. For each $(j, t, z) \in [\ell] \times \mathcal{T} \times [4]$, we add a constraint $x^z_{t,j} = x^z_{t,j-1} + \sum_{z' \in [4], t' \in \mathcal{T}} y^{z',z}_{(t',j-1),(t,j)} - \sum_{z' \in [4], t' \in \mathcal{T}} y^{z,z'}_{(t,j-1),(t',j)}$. These constraints ensure that the number of alternatives in each group is as described after the exchanges. This adds $4 \cdot 2^n \cdot \ell$ constraints.

6. For each $j \in \{0, \dots, \ell\}$, we add constraint $\sum_{t \in \mathcal{T}, z \in [4]} x^z_{t,j} = k$. This constraint ensures that each committee of the reconfiguration path has size k. This adds $\ell + 1$ constraints.

7. For each $v \in V$ and $j \in [\ell]$ we add a constraint $x_{v,j} = \sum_{t\colon v \in A(t), z \in [4]} x^z_{t,j}$. These constraints link the number of alternatives of each type with the number of alternatives each voter approves of. This adds $n \cdot \ell$ constraints.

8. For every $j \in [\ell]$ we add a constraint $\sum_{v \in V} f(x_{v,j}) \geq \mathrm{sc}_{\mathrm{PAV}}(\mathcal{P}, W) - \delta_{\mathrm{s}}$. These constraints ensure that each committee in the reconfiguration has a high enough score. This adds ℓ constraints.

The number of variables and constraints are both $O(4^n \cdot \ell)$, and the function f consists of k linear parts, and hence this IPWSPLT can be solved in FPT-time with respect to $n + \ell$. Note that the constraints 3-8 are the same as the constraints used by Yang and Wang.

We now show that the constructed IPWSPLT feasibility instance is a yes-instance if and only if there exists a reconfiguration path of at most length ℓ between the given committees W and W'.

The if part is straightforward: Let $W_0, \dots, W_\ell$ be a (δ_c, δ_s)-reconfiguration path for $(\mathcal{P}, \lambda)$. For every $t \in \mathcal{T}, j \in \{0, \dots, \ell\}$, we set the variable $x^z_{t,j}$ to be the number of alternatives in W_j that are also in the set represented by X^z_t, i.e., $|W_j \cap X^z_t|$. We additionally set for every $z, z' \in [4]$ the variable $y^{z,z'}_{(t,j-1),(t',j)}$ to be the number of alternatives of type t that were replaced by alternatives of type t' between the committees W_{j-1} and W_j. Constraint 5 is thus satisfied by construction. As these variables describe a valid committee, the constraints 6-8 are satisfied, as shown by Yang and Wang. Constraint 1 is satisfied, as $W_0 = W$ and $W_\ell = W'$. Constraint 2 is satisfied because the symmetric difference is upper bounded by $2\,\delta_c$. Constraint 3 must be satisfied, as it is not possible to have more alternatives of any type in the solution than there are in total. Similarly, constraint 4 must be satisfied, as we can only switch out alternatives from groups that were previously in the solution.

The only-if part follows similarly: We iteratively create the committees W_j for every $j \in [\ell]$. Note that $W_0 = W$ and $W_\ell = W'$ are already fixed due to constraint 1. Suppose the committees $W_1, \dots, W_j$ have been created and fulfill our conditions. We then create W_{j+1}. For each variable $y^{z,z'}_{(t,j),(t',j+1)}$, we replace that many alternatives that are in W_j and belong to the group X^z_t with alternatives that are not in W_j and belong to $X^{z'}_{t'}$. After this process, we have replaced at most δ_c many alternatives with δ_c many other alternatives due to Constraint 2, and therefore the symmetric difference is upper bounded by $2\,\delta_c$. The resulting committee must exist because Constraint 3 ensures that each group in each type of alternatives does not appear more often in a committee than possible, and Constraint 4 ensures that only existing alternatives were switched out. Due to constraint 5 $x^z_{t,j}$ is computed correctly. Constraint 6 ensures that the committee has size k. Constraint 7 counts the number of alternatives voters approve, and Constraint 8 ensures that the score is sufficiently high. As this reasoning holds for every $j \in [\ell]$, the generated $W_0, \dots, W_\ell$ is a (δ_c, δ_s)-reconfiguration path from W to W', as required.

Therefore, it is possible to find a path between two committees using the IPWSPLT feasibility formulation. □

The next results are based on either brute-force searching all possible reconfiguration paths or bounding the size of the reconfiguration graph.

Proposition 6.15. *For all score-based voting rules λ such that the score of a committee can be computed in polynomial time, the following holds:*

1. *λ-MR is* XP *with respect to ℓ for constant δ_c and can be solved in $m^{2\,\delta_c\,\cdot\ell}\cdot(n+m)^{O(1)}$ time.*

2. *If λ is additionally neutral, then λ-MR is* FPT *with respect to $n+k$ and* XP *with respect to n, and can be solved in $k^{2k}\cdot(2^n+2)^{2k}\cdot(n+m)^{O(1)}$ time and $(k+1)^{4\cdot 2^n}(n+m)^{O(1)}$ time; a shortest reconfiguration path can be found in the same time.*

Proof. For Statement 1, we observe that in each reconfiguration step we exchange at most δ_c alternatives with at most δ_c other alternatives. This gives us $\binom{m}{\delta_c}^2$ possible different exchanges. Since for a yes-instance we have at most ℓ steps in the reconfiguration path, we have $(\binom{m}{\delta_c}^2)^\ell = m^{2\,\delta_c\,\cdot\ell}$ possible different reconfiguration sequences. Since the score of a committee can be computed in polynomial time, we can check for every sequence whether it is a (δ_c,δ_s)-reconfiguration path from W to W' in $m^{2\,\delta_c\,\cdot\ell}\cdot(n+m)^{O(1)}$ time.

The proof idea for Statement 2 is to group alternatives of the same type, keep only a bounded number of alternatives for each type, and then generate a reconfiguration graph of bounded size. To this end, we enumerate the types as $1,\dots,2^n$. Let $\mathcal{T}$ denote the set of types of alternatives that exist in a given preference profile $\mathcal{P}$. We define C_t to be the set consisting of all alternatives with type t for $t\in\mathcal{T}$. We will analyze the running times one by one: Generally, let $\mathcal{P}=(\mathcal{C},\mathcal{V},\mathcal{R})$ be a preference profile and $I=(\mathcal{P},k,\delta_c,\delta_s,W,W')$ an instance of λ-MR.

For the first part of Statement 2, we observe that every committee on a reconfiguration path contains at most k alternatives of each type. Therefore, apart from the alternatives that appear in $W\cup W'$, we may remove all but k many alternatives of each type from the instance. Let us call the set of kept alternatives $\hat{V}$. This means that we only need to keep at most $2k+k\cdot 2^n = k(2^n+2)$ alternatives. We can construct a restricted reconfiguration graph of the remaining alternatives and determine a shortest path between u_W and $u_{W'}$ in time $(\binom{k(2^n+2)}{k})^2\cdot(n+m)^{O(1)} = (k(2^n+2))^{2k}\cdot(n+m)^{O(1)}$.

A u_W-$u_{W'}$ path of length ℓ' in the restricted reconfiguration graph corresponds to a (δ_c,δ_s)-reconfiguration path of length ℓ', as the committees corresponding to the vertices on the path form a (δ_c,δ_s)-reconfiguration path in I. Next assume we have a (δ_c,δ_s)-reconfiguration path $(W_0,\dots,W_{\ell'})$, where $W=W_0$ and $W'=W_{\ell'}$, of length ℓ' in I. We will show that there is a u_W-$u_{W'}$ path of length at most ℓ' in the restricted reconfiguration graph. Let us iteratively construct the path $u_{\hat{W}_1},\dots,u_{\hat{W}_{\ell'}}$. We show that in each step the following two conditions hold for $j\in\{0,\dots,\ell'\}$:

(C1) $W_j\cap(W\cup W') = \hat{W}_j\cap(W\cup W')$, i.e., an alternative from $W\cup W'$ is in W_j if and only if it is in $\hat{W}_j$, and

(C2) for every alternative type $t\in\mathcal{T}$, it holds that $|(W_j\cap\mathsf{C}_t)\setminus(W\cup W')| = |(\hat{W}_j\cap\mathsf{C}_t)\setminus(W\cup W')|$, i.e., for every type, the number of alternatives of that type is the same in both W_j and $\hat{W}_j$, discounting the alternatives in the initial and goal committees.

From these two conditions it follows clearly that $\text{SC}(\hat{W}_j) = \text{SC}(W_j)$ for every $j \in \{0, \dots, \ell'\}$. Since all alternatives in $W \cup W'$ are retained, the first vertex in the path is u_W and the last $u_{W'}$. Assume we have created vertices $u_{\hat{W}_0}, \dots, u_{\hat{W}_j}$ so that $\hat{W}_0, \dots, \hat{W}_j$ satisfy the conditions. We construct the committee $\hat{W}_{j+1}$ from $\hat{W}_j$ as follows:

1. We remove the alternatives in $(W_j \setminus W_{j+1}) \cap (W \cup W')$.
2. We add the alternatives in $(W_{j+1} \setminus W_j) \cap (W \cup W')$.
3. For every type $t \in \mathcal{T}$ such that $d \coloneqq |(W_j \cap \mathsf{C}_t) \setminus (W \cup W')| - |(W_{j+1} \cap \mathsf{C}_t) \setminus (W \cup W')| > 0$, we remove arbitrary d alternatives of type t from $\hat{W}_j \setminus (W \cup W')$.
4. For every type $t \in \mathcal{T}$ such that $d \coloneqq |(W_{j+1} \cap \mathsf{C}_t) \setminus (W \cup W')| - |(W_j \cap \mathsf{C}_t) \setminus (W \cup W')| > 0$, we add arbitrary d alternatives of type t from $(\hat{V} \cap \mathsf{C}_t \setminus \hat{W}_j) \setminus (W \cup W')$.

We show that these exchanges, if possible to perform, enforce that $\hat{W}_{j+1}$ satisfies the two conditions (C1) and (C2) we set to prove. The first two exchanges guarantee (C1), as they remove and add the corresponding alternatives to $\hat{W}_{j+1}$. The last two guarantee (C2), as they remove and add alternatives of the same type, as those exchanged between W_j to W_{j+1}. It is obvious that Exchanges 1 and 2 can be performed. Similarly, Exchange 3 is clearly possible as we remove alternatives that are necessarily in the committee. To see that we can always perform Exchange 4, recall that $\hat{V}$ contains $\hat{d}_t \coloneqq \min(k, |\mathsf{C}_t \cap \mathcal{C} \setminus (W \cup W')|)$ alternatives of type t. No committee may contain more than $\hat{d}_t$ many alternatives of $\mathsf{C}_t \setminus (W \cup W')$, so this exchange is also safe. It is also clear we will always add at most $|W_j \setminus W_{j+1}|$ alternatives and remove at most $|W_{j+1} \setminus W_j|$ alternatives, so $\hat{W}_j$ and $\hat{W}_{j+1}$ are δ_c-adjacent, and thus $u_{\hat{W}_j}$ and $u_{\hat{W}_{j+1}}$ are adjacent. This concludes the proof of the first part of Statement 2.

For the second part of Statement 2, we observe that there can be at most k alternatives of each type in a committee, so we can upper-bound the number of committees we need to track to $(k+1)^{2 \cdot 2^n}$: We associate each size-k committee $U \subseteq \mathcal{C}$, with two length-2^n vectors with non-negative entries such that for all $U \subseteq \mathcal{C}, t \in \mathcal{T}$,

$$f(U)[t] \coloneqq |(\mathsf{C}_t \cap U) \setminus W'| \text{ and } g(U)[t] \coloneqq |(\mathsf{C}_t \cap U) \cap W'|.$$

The vector $f(U)[t]$ is the number of alternatives of type t in the committee without W', and $g(U)[t]$ is the number of alternatives of type t in the committee that are also in W'. For these pairs of vectors we define δ_c-adjacency in the following way: Let $(f(W_1), g(W_1))$ and $(f(W_2), g(W_2))$ be the pairs of vectors representing two committees W_1 and W_2. We say that $(f(W_1), g(W_1))$ and $(f(W_2), g(W_2))$ are δ_c-adjacent if $\sum_{t \in \mathcal{T}} |f(W_1)[t] - f(W_2)[t]| + |g(W_1)[t] - g(W_2)[t]| \leq 2\,\delta_c$.

Clearly, there are at most $(k+1)^{2 \cdot 2^n}$ possible ordered pairs of vectors f and g. For each such ordered pair of vectors, we consider it *relevant* if there is a corresponding size-k committee, i.e., there are enough alternatives of each type in both W' and $\mathcal{C} \setminus W'$ and the score of such

a committee is at least $\text{SC}(W) - \delta_s$. Note that the scores of any two committees with the same corresponding ordered pairs of vectors must be equal because we are considering a neutral voting rule. Let $\hat{V} \coloneqq \{(f(U), g(U)) \mid U \subseteq \mathcal{C}, \text{SC}(U) \geq \text{SC}(W) - \delta_s, |U| = k\}$ be the set of ordered pairs of vectors that are relevant. We construct a restricted reconfiguration graph $\hat{G}$ whose vertices are $\hat{V}$, where there is an edge between two pairs of vectors if and only if they are δ_c-adjacent. Such a graph takes $(k+1)^{4 \cdot 2^n}(n+m)^{O(1)}$ time to construct.

We proceed to show that there is a (δ_c, δ_s)-reconfiguration path of length ℓ' between W and W' if and only if there is a path in $\hat{G}$ between $(f(W), g(W))$ and $(f(W'), g(W'))$ of length at most ℓ'.

Assume there is a (δ_c, δ_s)-reconfiguration path $(W_0, \ldots, W_{\ell'})$, where $W_0 = W$ and $W_{\ell'} = W'$, of length ℓ'. For every $W_j, j \in \{0, \ldots, \ell'\}$, we construct $(f(W_j), g(W_j))$ as previously described. We know that $(f(W_j), g(W_j))$ must be contained in $\hat{V}$ by construction. It is easy to observe that if $|W_j \Delta W_{j+1}| \leq 2\,\delta_c$, then $\sum_{t \in \mathcal{T}} |f(W_j)[t] - f(W_{j+1})[t]| + |g(W_j)[t] - g(W_{j+1})[t]| \leq 2\,\delta_c$ and the two vectors are δ_c-adjacent. Therefore, there exists a path in $\hat{G}$ between $(f(W), g(W))$ and $(f(W'), g(W'))$ of length at most ℓ'.

Assume there is a path in $\hat{G}$ between $(f(W), g(W))$ and $(f(W'), g(W'))$ of length at most ℓ'. We construct a reconfiguration path $(W_0, \ldots, W_{\ell'})$ such that $W_0 = W$, $W_{\ell'} = W'$ and for every $j \in \{0, \ldots, \ell'\}, t \in \mathcal{T}$, it holds that $f(W_j)[t] = |(\mathsf{C}_t \cap W_j) \setminus W'|$ and $g(W_j)[t] = |(\mathsf{C}_t \cap W_j) \cap W'|$.

We set $W_0 \coloneqq W$. Assuming we have constructed committees $W_0, \ldots, W_j$ for some $j \in \{0, \ldots, \ell' - 1\}$ so that the above conditions are satisfied, we construct the next committee W_{j+1} as follows:

1. For every type $t \in \mathcal{T}$, if $f(W_{j+1})[t] - f(W_j)[t] > 0$, then we add arbitrary $f(W_{j+1})[t] - f(W_j)[t]$ many alternatives from $(\mathsf{C}_t \setminus W_j) \setminus W'$ to W_j.
2. For every type $t \in \mathcal{T}$, if $f(W_j)[t] - f(W_{j+1})[t] > 0$, then we remove arbitrary $f(W_j)[t] - f(W_{j+1})[t]$ many alternatives from $(W_j \cap \mathsf{C}_t) \setminus W'$ from W_j.
3. For every type $t \in \mathcal{T}$, if $g(W_{j+1})[t] - g(W_j)[t] > 0$, then we add arbitrary $g(W_{j+1})[t] - g(W_j)[t]$ many alternatives from $(\mathsf{C}_t \setminus W_j) \cap W'$ to W_j.
4. For every type $t \in \mathcal{T}$, if $g(W_j)[t] - g(W_{j+1})[t] > 0$, then we remove arbitrary $g(W_j)[t] - g(W_{j+1})[t]$ many alternatives from $(W_j \cap \mathsf{C}_t) \cap W'$ from W_j.

It is easy to see that if these exchanges can be performed, then $f(W_{j+1})[t] = |(\mathsf{C}_t \cap W_{j+1}) \setminus W'|$ and $g(W_{j+1})[t] = |(\mathsf{C}_t \cap W_{j+1}) \cap W'|$ must hold. We can perform Exchanges 1 and 3, as per construction, there must be enough alternatives of the selected type t in $(W_j \cap \mathsf{C}_t) \setminus W'$ and $(W_j \cap \mathsf{C}_t) \cap W'$. Similarly, exchanges 2 and 4 can be performed, because there must be sufficiently many alternatives in $(\mathsf{C}_t \setminus W') \setminus W_j$ and $(\mathsf{C}_t \cap W') \setminus W_j$, because otherwise W_{j+1} could not exist in the reconfiguration path. Additionally, because $\vec{W}_j$ and $\vec{W}_{j+1}$ are δ_c-adjacent, the committees W_j and W_{j+1} must also be

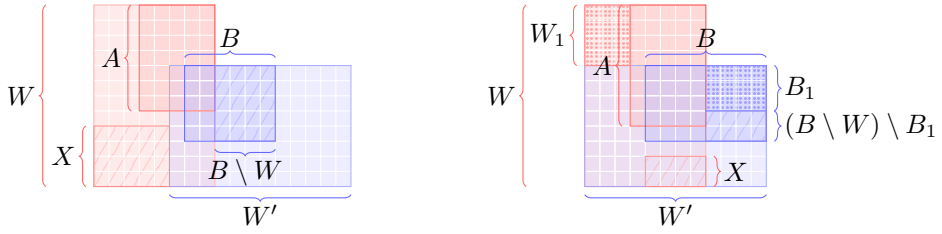


Figure 6.2: Illustration for the proof of Lemma 6.16, depicting how the two committees W and W' overlap and the blocks A, B, W_1, B_1, and X. Left: Case 1; right: Case 2.

δ_c-adjacent, as the number of additions and removals in 1–4 is upper-bounded by $\sum_{t\in\mathcal{T}} |f(W_j)[t] - f(W_{j+1})[t]| + |g(W_j)[t] - g(W_{j+1})[t]| \leq 2\,\delta_c$. The score of W_{j+1} must be sufficient because otherwise $(f(W_{j+1}), g(W_{j+1}))$ would not be on the path.

To see that $W_{\ell'} = W'$, it is enough to observe that W' is the only committee $U \subseteq \mathcal{C}$ such that $f(U)[t] = |(\mathsf{C}_t \cap W') \setminus W'| = 0$ and $g(U)[t] = |(\mathsf{C}_t \cap W') \cap W'| = |\mathsf{C}_t \cap W'|$ for every $t \in \mathcal{T}$. Thus we have that $W_{\ell'} = W'$.

The FPT (resp. XP) statement follows directly from the time of constructing $\hat{G}$ and searching for a shortest path. □

For CC, the following lemma shows that for a large enough committee size k there is always a reconfiguration path: The score of each committee is determined by at most n relevant alternatives since each voter contributes a score of at most one. Due to this, we can find a shortest path if the symmetric difference between W and W' is very large (Case 1), by only exchanging the alternatives in the symmetric difference. Otherwise (Case 2), we can still find a path in polynomial time by also exchanging alternatives that are shared by both W and W'.

Lemma 6.16. *For every instance of CC-MR with $k \geq 2n$ there exists a reconfiguration path between W and W'. Moreover,*

1. *If $|W \setminus W'| \geq 2n$, then we can find a shortest one in polynomial time.*
2. *If $|W \setminus W'| < 2n$, then we can find one with length at most $\lceil \frac{2n}{\delta_c} \rceil$ in polynomial time.*

Proof. Let $I = (\mathcal{P}, k, \delta_c, \delta_s, W, W')$ denote an instance of CC-MR. We first consider the case when $\delta_c = 1$ and show how to adapt the proof for $\delta_c \geq 2$ at the end. First, observe that we can always find a size-n committee of W (resp. W') with score at least $\textsc{sc}(W)$ (resp. $\textsc{sc}(W')$). Hence, the idea is to find a size-n committee $A \subseteq W$ (resp. $B \subseteq W'$) such that $\textsc{sc}(A) = \textsc{sc}(W)$ (resp. $\textsc{sc}(B) = \textsc{sc}(W')$) and perform exchanges to obtain a committee that contains $A \cup B$, and then switch out alternatives that are not contained in W'. Note that A and B can be found in polynomial time by iteratively

deleting "redundant" alternatives from W and W', respectively. To show the statement, we distinguish between two cases. Figure 6.2 illustrates the constructions used in the proofs.

Case 1: $|W \setminus W'| \geq 2n$ or $|W \setminus (W' \cup A)| \geq |B \setminus W|$. Note that the first condition implies the second condition since $|A| = n = |B|$. To implement the exchange idea, let $X \subseteq W \setminus (W' \cup A)$ be an arbitrary committee of $|B \setminus W|$ alternatives. We divide the exchanges into two parts. In the first part, we step by step exchange all alternatives from X with all alternatives from $B \setminus W$. It is straightforward to verify that every committee obtained until now has a score of at least $\text{SC}(W)$ because it always contains A, and the last committee obtained has a score of at least $\text{SC}(W')$ since it contains B. In the second part, we step-by-step exchange the remaining alternatives from $W \setminus (W' \cup X)$ with the remaining alternatives from $W' \setminus (W \cup B)$. It is straightforward to verify that every committee obtained in the second part has score at least $\text{SC}(W') \geq \text{SC}(W) - \delta_{\text{s}}$ since it contains B. The exchange sequence yields a reconfiguration path of shortest length $\lceil |W \Delta W'|/2 \rceil$. If $\delta_{\text{c}} \geq 2$, then we take every δ_{c}-th element on the path, including W and W'. All these committees have score at least $\text{SC}(W) - \delta_{\text{s}}$ by construction, and the path is of the shortest possible length $\lceil |W \Delta W'|/(2\,\delta_{\text{c}}) \rceil$.

Case 2: $|W \setminus W'| < 2n$ and $|W \setminus (W' \cup A)| < |B \setminus W|$. We proceed in three parts. For the sake of brevity, let $W_1 = W \setminus (W' \cup A)$ and let $B_1 \subseteq B \setminus W$ denote a committee of $|W_1|$ alternatives arbitrarily chosen from $B \setminus W$. First, we step-by-step exchange the alternatives in W_1 with the alternatives in B_1. Afterwards, we select a committee $X \subseteq (W \cap W') \setminus (A \cup B)$ of $|B \setminus W| - |W_1|$ arbitrary alternatives, and step-by-step exchange the alternatives in X with the remaining alternatives in $(B \setminus W) \setminus B_1$. Note that this is possible since $|(W \cap W') \setminus (A \cup B)| \geq |W \cap W'| - |A| + |A \setminus W'| - |B| + |B \setminus W| = |W| - |W \setminus W'| - |A| + |A \setminus W'| - |B| + |B \setminus W| \geq -|W_1| + |B \setminus W|$; the last inequality holds since $|W_1| = |W \setminus (W' \cup A)| = |W \setminus W'| - |A \setminus W'|$ and $|W| \geq 2n \geq |A| + |B|$. So far, we have performed $|B \setminus W| \leq n$ exchanges, where all intermediate committees contain A, and the final committee additionally contains B. Thus, they all have a score of at least $\text{SC}(W)$. The only alternatives from $W \setminus W'$ that remain in the most recently obtained committee are those in $A \setminus W'$. Thus, in the third part, we step-by-step exchange all alternatives in $A \setminus W'$ with the alternatives in W' that are not yet in the most recent committee; they are precisely in $X \cup (W' \setminus (W \cup B))$. Since all committees obtained in the third part contain B, each of them has score $\text{SC}(W') \geq \text{SC}(W) - \delta_{\text{s}}$. The number of exchanges in the last part is $|A \setminus W'| \leq n$, and overall we have performed at most $2n$ exchanges. Similarly to the first case, if $\delta_{\text{c}} \geq 2$, then the number of exchanges reduces to $\lceil 2n/\,\delta_{\text{c}} \rceil$. □

Combining Lemma 6.16 with Proposition 6.15.2 yields an FPT result (with respect to n) for CC.

Theorem 6.17. *CC-MR is* FPT *with respect to n, and a shortest reconfiguration path can be found in* FPT*-time with respect to n.*

Proof. We distinguish between two cases. If $k < 2n$, then we use Proposition 6.15.2 to solve CC-MR and find a shortest path in FPT-time with respect to n. Otherwise, by Lemma 6.16, we can in polynomial time find a reconfiguration path that either has minimum length (when $|W \setminus W'| \geq 2n$) or has length at most $\lceil \frac{2n}{\delta_c} \rceil$ (when $|W \setminus W'| < 2n$). This proves the first part of the statement.

To show the second part of the statement, we only need to consider the case when $k \geq 2n$ and $|W \setminus W'| < 2n$ as the other cases are already covered by the above reasoning and by Lemma 6.16.1. In the case of $k \geq 2n$ and $|W \setminus W'| < 2n$, by Lemma 6.16.2, there is always a reconfiguration path of length at most $\lceil \frac{2n}{\delta_c} \rceil$. This implies that a shortest path can exchange at most $4n$ many alternatives. Since CC is neutral, any two alternatives of the same type are "interchangeable" on the path. Hence, we only need to keep track of at most $4n$ alternatives for each type of alternative. Note that all alternatives in the symmetric difference $W \Delta W'$ must be among these tracked alternatives. We note that we can assume $\delta_c < 2n$, as otherwise there is a path of length one that is obviously a shortest path. Recall that due to CC being neutral, it suffices to keep track of $4n$ alternatives of each type since by Lemma 6.16.2 the maximum number of alternatives that are exchanged on a shortest path is $4n$. Let X denote a set consisting of $\min(4n, |\mathsf{C}_t|)$ alternatives of each type $t \in \mathcal{T}$, including all alternatives from $W \Delta W'$. This implies that $|X| \leq 4n2^n$. As we know that the reconfiguration path has length at most $\ell \leq \lceil 2n/\,\delta_c \rceil$, we can solve this problem by branching over all possible exchanges using alternatives in X with depth ℓ. For each step $i \in [\ell]$, we branch over all pairs of two sets $X_i^1, X_i^2 \subseteq X$ with $|X_i^1| \leq \delta_c$ and $|X_i^2| \leq \delta_c$ that shall correspond to the set of alternatives in $W_{i-1} \setminus W_i$ and $W_i \setminus W_{i-1}$, respectively. Then, using these exchanges, we can compute all committees on each branch and check whether they yield a desired reconfiguration path. By checking all branches we can then find a shortest path.

It remains to analyze the running time. In the case where $k < 2n$, the time needed to compute a shortest path is still $(2n)^{4n} \cdot (2^n + 2)^{4n} \cdot (n+m)^{O(1)}$ per Proposition 6.15.2. For $k \geq 2n$ and $|W \setminus W'| \geq 2n$ a shortest path can be found in polynomial time per Lemma 6.16.1.

For the case of $k \geq 2n$ and $|W \setminus W'| < 2n$, we compute the running time of the branching algorithm. A single step has $|X|^{\delta_c} \cdot |X|^{\delta_c} = |X|^{2\,\delta_c}$ many branches and there are at most $\lceil \frac{2n}{\delta_c} \rceil$ steps. That means the branching algorithm has a running time of $|X|^{2\,\delta_c\,\ell} \leq |X|^{2\,\delta_c \lceil \frac{2n}{\delta_c} \rceil} \leq |X|^{4n+2\,\delta_c} \leq (4n \cdot 2^n)^{8n}$. □

By observing that there are not too many different alternatives (with respect to $n + \mathsf{b}$), we obtain the following FPT result.

Proposition 6.18. *For all neutral and score-based multi-winner voting rules λ such that the score of a committee can be computed in polynomial time, λ-MR is* FPT *with respect to $n + \mathsf{b}$ and can be solved in $2^{2\mathsf{b}n}(n+m)^{O(1)}$ time.*

Proof. Let $\mathcal{P} = (\mathcal{C} = [m], \mathcal{V}, \mathcal{R})$ be a preference profile and $I = (\mathcal{P}, k, \delta_c, \delta_s, W, W')$ an instance of λ-MR. As every voter approves at most b alternatives, the number of alternatives that are approved by at least one voter is at most $\mathsf{b}n$. Let $\hat{\mathcal{C}} = \cup_{i \in \mathcal{V}} \mathsf{A}_i$ denote the set of alternatives that are approved by at least one voter. Then $|\hat{\mathcal{C}}| \leq \mathsf{b}n$. We call these alternatives *relevant* and the rest of the alternatives *irrelevant*.

For every two size-k committees W_1, W_2 such that $W_1 \cap \hat{\mathcal{C}} = W_2 \cap \hat{\mathcal{C}}$, there is a reconfiguration path from W_1 to W_2 by swapping the irrelevant alternatives one by one. This is true because the irrelevant alternatives are not approved by any voter, and λ is neutral, so their identity does not affect the score of the committee. We can use this observation to create a modified reconfiguration graph that disregards information about the identities of irrelevant alternatives.

We create a modified reconfiguration graph $\hat{G^{\mathsf{R}}}$. The vertices are all at most size-k committees of $\hat{\mathcal{C}}$ such that, possibly together with some irrelevant alternatives, they could be committees on a reconfiguration path. Formally, the set of vertices of $\hat{G^{\mathsf{R}}}$ is

$$V(\hat{G^{\mathsf{R}}}) = \{u_{\mathcal{C}'} \mid \mathcal{C}' \subseteq \hat{\mathcal{C}} \text{ s.t. } |\mathcal{C}'| \leq k \wedge \mathrm{sc}_\lambda(W) - \mathrm{sc}_\lambda(\mathcal{C}' \cup \hat{W}) \leq \delta_s \text{ for some } \hat{W} \subseteq \mathcal{C} \setminus \hat{\mathcal{C}} \text{ with } |\hat{W}| = k - |\mathcal{C}'|\}.$$

Since $|\hat{\mathcal{C}}| \leq \mathsf{b}n$, we have that $|V(\hat{G^{\mathsf{R}}})| \leq 2^{\mathsf{b}n}$. There is an edge between two vertices u_{W_1} and u_{W_2} if there are two committees containing W_1 and W_2 as their relevant alternatives, which are δ_c-adjacent. Formally u_{W_1} and u_{W_2} are adjacent if $|W_1 \setminus W_2| \leq \delta_c$ and $|W_2 \setminus W_1| \leq \delta_c$. The number of edges of $\hat{G^{\mathsf{R}}}$ is bounded above by $|V(\hat{G^{\mathsf{R}}})|^2$ and therefore $(2^{\mathsf{b}n})^2$. As a shortest path between two vertices can be determined in time linear in the number of edges of the graph, we can search for a path from $u_{W \cap \hat{\mathcal{C}}}$ to $u_{W' \cap \hat{\mathcal{C}}}$ in $2^{2\mathsf{b}n} \cdot (n+m)^{O(1)}$ time.

Finally, we show that there exists a path from $u_{W \cap \hat{\mathcal{C}}}$ to $u_{W' \cap \hat{\mathcal{C}}}$ in the modified reconfiguration graph if and only if there exists a (δ_c, δ_s)-reconfiguration path for I.

For the "if" direction, let $W_0, \ldots, W_t$ with $W_0 = W$ and $W_t = W'$ be a reconfiguration path. For each committee W_i on the path, there must exist a vertex $u_{W_i \cap \hat{\mathcal{C}}}$. Since $|W_i \setminus W_{i+1}| \leq \delta_c$ and $|W_{i+1} \setminus W_i| \leq \delta_c$ it follows that $|(W_{i+1} \cap \hat{\mathcal{C}}) \setminus (W_i \cap \hat{\mathcal{C}})| \leq \delta_c$ and $|(W_i \cap \hat{\mathcal{C}}) \setminus (W_{i+1} \cap \hat{\mathcal{C}})| \leq \delta_c$. Therefore it follows that $u_{W_1 \cap \hat{\mathcal{C}}}, \ldots, u_{W_t \cap \hat{\mathcal{C}}}$ is a path in the modified reconfiguration graph.

For the "only if" direction, we first note that irrelevant alternatives can be exchanged with other irrelevant alternatives without changing the score. We order the irrelevant alternatives in an arbitrary but fixed way such that the irrelevant alternatives in W are before all other irrelevant alternatives. Let $u_{V_0}, \ldots, u_{V_t}$ be a path in the reconfiguration graph such that $V_0 = W \cap \hat{\mathcal{C}}$ and $V_t = W' \cap \hat{\mathcal{C}}$. We will now iteratively build a reconfiguration path. We set $W_0 = W$. Let $W_0, \ldots, W_i$ be the committees corresponding to the first i vertices on the path. We create W_{i+1} in the following way:

If $|V_i| = |V_{i+1}|$, then we set $W_{i+1} = W_i \setminus (V_i \setminus V_{i+1}) \cup (V_{i+1} \setminus V_i)$. In this case, $|W_{i+1}| = |W_i|$, $|W_{i+1} \setminus W_i| = |V_{i+1} \setminus V_i| \leq \delta_c$, and $|W_i \setminus W_{i+1}| = |V_i \setminus V_{i+1}| \leq \delta_c$. Therefore, this is a valid reconfiguration step.

If $|V_i| < |V_{i+1}|$, then let I be the $|V_{i+1}| - |V_i|$ many irrelevant alternatives with the highest index that are in W_i. We then set $W_{i+1} = (W_i \setminus I) \setminus (V_i \setminus V_{i+1}) \cup (V_{i+1} \setminus V_i)$. In this case $|W_{i+1}| = |W_i|$ and due to this $|W_{i+1} \setminus W_i| = |W_i \setminus W_{i+1}|$. As $|W_{i+1} \setminus W_i| = |V_{i+1} \setminus V_i| \leq \delta_c$, this is also a valid reconfiguration step.

In the last case that $|V_i| > |V_{i+1}|$, let I be the $|V_i| - |V_{i+1}|$ many irrelevant alternatives with the lowest index that are not in W_i. We then set $W_{i+1} = W_i \setminus (V_i \setminus V_{i+1}) \cup (V_{i+1} \setminus V_i) \cup I$. In this case $|W_{i+1}| = |W_i|$ and due to this $|W_{i+1} \setminus W_i| = |W_i \setminus W_{i+1}|$. As $|W_i \setminus W_{i+1}| = |V_i \setminus V_{i+1}| \leq \delta_c$, this is also a valid reconfiguration step.

It is easy to see that the obtained W_{i+1} is in the reconfiguration graph, as $\text{SC}(V_{i+1} \cup (W_{i+1} \setminus \hat{\mathcal{C}})) = \text{SC}(W_{i+1}) \geq \text{SC}(W) - \delta_s$.

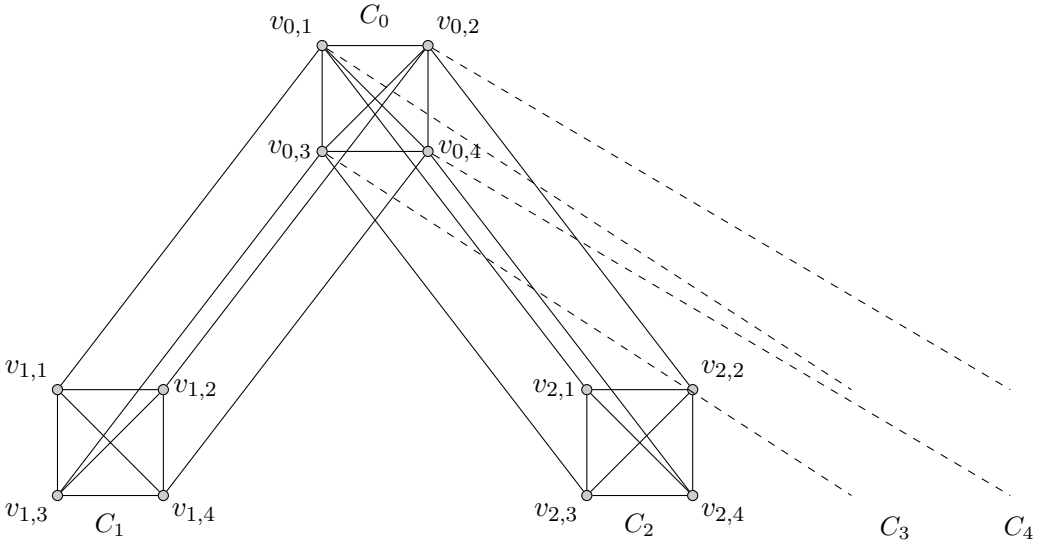
By repeating this process, we get a reconfiguration path from W to a committee W_t which satisfies $W_t \cap \hat{\mathcal{C}} = W' \cap \hat{\mathcal{C}}$. Therefore, the alternatives in $W_t \Delta W'$ are only irrelevant alternatives. We can greedily exchange these alternatives to obtain a reconfiguration path from W_t to W', thereby completing our proof. □

Finally, we prove that the committee size k combined with the path length ℓ is unlikely to yield FPT-algorithms by showing W[2]-hardness. We reduce from the problem of Dominating Set Reconfiguration via Token Jumping (DSR-TJ). As a reminder, in DSR-TJ, we are given a graph $G = (V, E)$, two non-negative integers h and $\hat{\ell}$, and two size-h dominating sets V_s and V_t of G, and we ask whether there exists a sequence $V_0, \ldots, V_{\hat{\ell}}$ of size-h dominating sets such that $V_0 = V_s$, $V_{\hat{\ell}} = V_t$ and $|V_i \Delta V_{i+1}| \leq 2$ for all $0 \leq i \leq \hat{\ell} - 1$. Recall that a dominating set of a graph is a subset V' of vertices whose neighborhood covers the whole vertex set of the graph, i.e., each vertex not from V' is adjacent to at least one vertex in V'. Bodlaender et al. [BGS26] claimed that Mouawad et al. [MNR+17] showed that DSR-TJ is W[2]-hard with respect to $(h, \hat{\ell})$, where h denotes the size of any dominating set (DS for short) on the path and $\hat{\ell}$ the length of a shortest DS reconfiguration path (if it exists). The reduction given by Mouawad et al., however, only works for the TAR but not the TJ variant. The reason is that in their created instance, the size h_0 of the initial DS is smaller than h, so the size of any intermediate dominating set could be larger than h_0, while this is not possible for the TJ variant. Below, we modify their reduction and show hardness for the TJ variant.

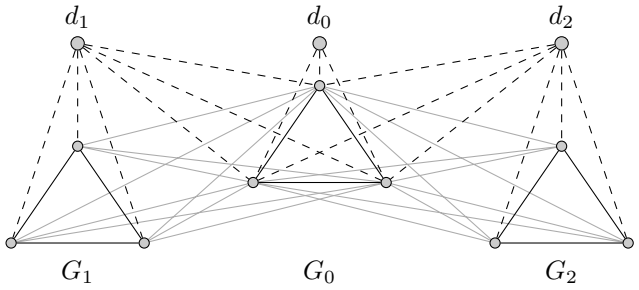
Proposition 6.19. *DSR-TJ is W[2]-hard with respect to $h + \hat{\ell}$.*

Proof. We provide a parameterized reduction from the W[2]-complete Dominating Set problem[6], which is adapted from one of Mouawad et al. [MNR+17]. We will point out the differences in the reduction.

[6] See Section 2.7 for a definition and additional information.



(a) The graph H_1 for $k_{\mathrm{DS}} = 1$: five size-4 cliques $C_0, \ldots, C_4$ with each vertex in cliques $C_1, \ldots, C_4$ having an edge to the same-index vertex in C_0 (only those in C_0, C_1, and C_2 are drawn).



(b) The graph H_2 for $k_{\mathrm{DS}} = 1$ with $\hat{G}$ being the triangle with 3 vertices: three copies G_0, G_1, G_2 of $\hat{G}$, with each vertex in G_1 and G_2 being connected to the matching vertex in G_0 and its neighbors. The vertices d_0, d_1, d_2 have an edge to every vertex in G_0. Finally, d_1 has an edge to every vertex in G_1 and d_2 has an edge to every vertex in G_2.

Figure 6.3: Construction from Proposition 6.19 for $k_{\mathrm{DS}} = 1$ and $\hat{G} = K_3$, the complete graph with three vertices, i.e., a triangle.

We first show the construction and then the correctness. Note that in our construction, the size of the dominating set as well as the length of a shortest reconfiguration path in the reduced instance will be linearly bounded by k_{DS}.

Construction. Let $(\hat{G}, k_{\mathrm{DS}})$ be a DS instance with $\hat{V} = \{u_1, \ldots, u_{\hat{n}}\}$ being the vertex set of $\hat{G}$. The graph G of the created instance of DSR-TJ consists of two independent graphs H_1 and H_2.

The first graph H_1 consists of $k_{\mathrm{DS}}+4$ cliques (i.e., complete subgraphs), each of size $k_{\mathrm{DS}}+3$ (rather than $k_{\mathrm{DS}}+2$ cliques consisting of $k_{\mathrm{DS}}+1$ vertices as in the original reduction of Mouawad et al.). We refer to these cliques as $C_0, \ldots, C_{k_{\mathrm{DS}}+3}$. We furthermore label the vertices in each clique C_i as $v_{i,1}, \ldots, v_{i,k_{\mathrm{DS}}+3}$. Finally, we add to H_1 the following edges $E' = \{\{v_{0,i}, v_{z,i}\} \mid \forall\, i, z \in [k_{\mathrm{DS}}+3]\}$. In other words, we make each vertex in C_0 adjacent to the vertex in each other clique C_z ($z \in [k_{\mathrm{DS}}+3]$) that has the same second index. Therefore, each vertex in each of the cliques $C_1, \ldots, C_{k_{\mathrm{DS}}+3}$ is adjacent to a vertex in C_0, with no two vertices from the same clique C_z being adjacent to the same vertex in C_0.

The second graph H_2 is the same as the one in the original reduction. It consists of $k_{\mathrm{DS}}+2$ copies of the original graph $\hat{G}$ which we label as $G_0 = (V_0, E_0), \ldots, G_{k_{\mathrm{DS}}+1} = (V_{k_{\mathrm{DS}}+1}, E_{k_{\mathrm{DS}}+1})$. For each $z \in \{0, \ldots, k_{\mathrm{DS}}+1\}$, let $w_{z,1}, \ldots, w_{z,|V|}$ denote the vertices in G_z. We add to H_2 the following edges $E'' = \{\{w_{0,i}, w_{z,j}\} \mid i, j \in [\hat{n}], z \in [k_{\mathrm{DS}}+1] \text{ s.t. } \{u_i, u_j\} \in E(\hat{G}) \vee i = j\}$. In other words, we make each vertex $w_{0,i} \in G_0$ adjacent to the copies of its neighbors in G_0 and to its copies $w_{1,i}, \ldots, w_{k_{\mathrm{DS}}+1,i}$ in $G_1, \ldots, G_{k_{\mathrm{DS}}+1}$.

Finally, we add to H_2 a set D of $k_{\mathrm{DS}}+2$ dummy vertices $d_0, \ldots, d_{k_{\mathrm{DS}}+1}$ and edges that connect the copied graphs, one for each copy. For every $z \in \{0, \ldots, k_{\mathrm{DS}}+1\}$, we make d_z adjacent to all vertices in G_z and all vertices in G_0. A sketch of the construction can be seen in Figure 6.3.

We define the initial and target dominating sets as $V_s = V(C_0) \cup D \cup \{w_{0,1}, \ldots, w_{0,k_{\mathrm{DS}}}\}$ and $V_t = \{v_{i,i} \mid i \in [k_{\mathrm{DS}}+3]\} \cup D \cup \{w_{0,1}, \ldots, w_{0,k_{\mathrm{DS}}}\}$, respectively. Note that the set $\{w_{0,1}, \ldots, w_{0,k_{\mathrm{DS}}}\}$ is not necessary to dominate the vertices in the created graph, but is needed to maintain the size bound; it can be replaced with an arbitrary but fixed vertex subset of $V(G_0)$ of size k_{DS}. We use this subset for ease of reasoning.

To complete the construction, we define the size bound $h = 3k_{\mathrm{DS}}+5$ and $\hat{\ell} = 4k_{\mathrm{DS}}+5$. Clearly, the construction can be done in polynomial time and $|V_s| = |V_t| = h$.

Correctness. We show that $(\hat{G}, k_{\mathrm{DS}})$ admits a size-k_{DS} dominating set if and only if the constructed instance has a length-$\hat{\ell}$ reconfiguration path between V_s and V_t. For the "only if" direction, let U' denote a size-k_{DS} dominating set of $\hat{G}$. We first exchange the k_{DS} vertices $w_{0,z}$, $z \in [k_{\mathrm{DS}}]$, one-by-one with the vertices from the copies corresponding to U', i.e., $\{w_{0,i} \mid u_i \in U'\}$. Afterwards, we exchange each dummy vertex from D with a distinct vertex $v_{z,z}$, $z \in [k_{\mathrm{DS}}+2]$. Next, we exchange one vertex from $V(C_0)$ with vertex $v_{k_{\mathrm{DS}}+3,k_{\mathrm{DS}}+3}$. Note that now the remaining vertices from the clique C_0 are not necessary to dominate the vertices in the created graph anymore. Thus, we can exchange the remaining $k_{\mathrm{DS}}+2$ vertices from the clique C_0 with the $k_{\mathrm{DS}}+2$ dummies from D, one-by-one. Finally, we exchange back the vertices in $\{w_{0,i} \mid u_i \in U'\}$ with the vertices in $\{w_{0,1}, \ldots, w_{0,k_{\mathrm{DS}}}\}$ one-by-one, reaching the target dominating set. The length of the reconfiguration path is $k_{\mathrm{DS}} + k_{\mathrm{DS}} + 2 + k_{\mathrm{DS}} + 3 + k_{\mathrm{DS}} = 4k_{\mathrm{DS}}+5 = \hat{\ell}$.

For the "if" direction, assume that the constructed instance admits a length-$\hat{\ell}$ reconfiguration path between V_s and V_t. We aim to show that on the reconfiguration path, there

must be a dominating set of $H_1 \cup H_2$ containing a size-k_{DS} subset which corresponds to a dominating set of $\hat{G}$. We recall an observation by Mouawad et al. that for $k_{\mathrm{DS}}+4$ cliques, each of size $k_{\mathrm{DS}}+3$, and connected as described in their construction, they need $k_{\mathrm{DS}}+3$ additional vertices on that graph in order to be able to switch from the dominating set consisting of $V(C_0)$ to the dominating set $\{v_{i,i} \mid i \in [k_{\mathrm{DS}}+2]\}$ via TAR, as one needs to dominate all vertices from C_z, $z \in [k_{\mathrm{DS}}+3]$, before removing any vertex from C_0.

We show similarly that for the TJ variant we can only afford to have k_{DS} vertices to dominate all vertices from H_2 in order to "save up" $k_{\mathrm{DS}}+2$ to exchange for the vertices from $\{v_{z,z} \mid z \in [k_{\mathrm{DS}}+2]\}$. To achieve this, we claim that for each dominating set X on the reconfiguration path which does not contain all vertices from the clique C_0, i.e., $|X \cap V(C_0)| \neq |V(C_0)|$, it must hold that $|V(C_z) \cap X| \geq 1$ for every $z \in [k_{\mathrm{DS}}+3]$. Let $v_{0,i}$ be a vertex that is not included in X. Then, in order to dominate all vertices from $\{v_{z,i} \mid z \in [k_{\mathrm{DS}}+3]\}$, we need a vertex from each clique C_z, as claimed. Since no vertex in C_0 is contained in V_t, in order to reconfigure from V_s to V_t, there must be an intermediate dominating set X' where one vertex from $V(C_0)$ is missing, and at least one vertex from each clique C_z is present. This means that X' contains $k_{\mathrm{DS}}+3+k_{\mathrm{DS}}+2 = 2k_{\mathrm{DS}}+5$ vertices from the first graph H_1. By the size bound $h = 3k_{\mathrm{DS}}+5$, we need k_{DS} vertices to dominate the second graph H_2. By construction, this is possible only if it corresponds to a size-k_{DS} dominating set of the input graph $\hat{G}$. This concludes the "if" direction and the correctness proof. □

We obtain the same hardness for the CC rule by adapting the reduction by Betzler et al. [BSU13]: Each vertex becomes an alternative and a voter such that voters approve of alternatives corresponding to themselves and adjacent vertices. Committees maximizing the CC-score then correspond to dominating sets.

Theorem 6.20. *CC-MR^0 (and hence CC-MR) is W[2]-hard with respect to $k+\ell$.*

Proof. It suffices to show W[2]-hardness for CC-MR^0. To achieve this, we provide a parameterized reduction from the W[2]-hard DSR-TJ problem (see Proposition 6.19) to CC-MR^0, i.e., CC-MR where $\delta_{\mathrm{s}} = 0$ and $\delta_{\mathrm{c}} = 1$.

Let $(G, h, \hat{\ell}, V_s, V_t)$ be an instance of DSR-TJ, where G is an undirected graph, V_s and V_t are the initial and target dominating sets, respectively, $h = |V_s| = |V_t|$, and $\hat{\ell}$ denotes the length bound on the reconfiguration path (if it exists).

We define $k = h$ and create a preference profile as follows. For each vertex $v_i \in V(G)$, create

- one *vertex-alternative* c_i, and
- one *vertex-voter* u_i that approves of c_i and the alternatives corresponding to the neighbors of v_i in G. Formally, the approval set of voter u_i is $\mathsf{A}_i = \{c_i\} \cup \{c_j \mid \{v_i, v_j\} \in E(G)\}$.

Let $\mathcal{P} = (\mathcal{C}, \mathcal{V}, \mathcal{R})$ denote the created profile. The total number of voters in this instance is $|V(G)|$. To complete the construction, let the initial and target committees be $W = \{c_i \mid v_i \in V_s\}$ and $W' = \{c_i \mid v_i \in V_t\}$. Note that the score of W (resp. W') is $|V(G)|$ since V_s (resp. V_t) is a dominating set. It is straightforward to see that every size-h dominating set of G corresponds to a size-k committee with score at least $|V(G)|$. The correctness of the construction follows immediately, as the bijection between dominating sets and winning committees preserves adjacency. □

6.4 Conclusion

In this chapter, we introduced a framework for reconfiguring committees in multi-winner elections for four prominent committee election rules: CC, PAV, AV, and SAV, and systematically studied the (parameterized) complexity of the problem. It remains as an open question whether our reconfiguration problem under PAV is FPT with respect to the number n of voters. Note that the approach for CC (Theorem 6.17) does not directly extend to PAV. The key step in that proof relies on the fact that, under CC, a committee of size n already achieves the maximum possible score, since each voter contributes at most 1 to the score. Under PAV, however, a voter's contribution depends on how many approved alternatives are in the committee, so no analogous size bound on a representative sub-committee exists.

Our framework can be directly applied to other score-based rules, such as the Monroe rule [Mon95], even for linear preferences. For non-score-based rules, we may adapt by either restricting the reconfiguration path to winning committees or introducing a measure that captures closeness to optimality.

The concept of reconfiguration extends beyond the multi-winner election settings explored here and the fair division setting explored by Igarashi et al. [IKSY24]. Many other social choice frameworks remain unexplored in this context, such as stable matching [GS62] or coalition formation [DG80]. Investigating these could yield insights into the algorithmic principles underlying varied decision-making processes.

CHAPTER 7

Conclusion

In this thesis, we studied four novel models related to well-known problems in collective decision-making from both parameterized and classical complexity perspectives. Even though intractability persisted through (almost) all topics, we were able to identify several islands of tractability. We considered two challenges:

1. How can agents reach a joint outcome under which cooperation is self-enforcing—one that no individual or coalition has reason to reject?
2. How can an existing outcome be reconfigured to a new one dynamically, through a sequence of small, controlled changes?

In Chapters 3 to 5, we tackled the first challenge. We used TU- and NTU-games to model two different settings: those with transferable utility and those with non-transferable utility. We also considered two notions of stability: individual rationality to model only individual deviation, and the stronger notion core stability to model also coalitional deviations. The table below shows which notion we studied in which chapter:

	TU	NTU
IR	Chapter 3	Chapter 4
Core	Chapters 3 and 5	Chapter 5

Note that we mentioned individual rationality only briefly in Chapter 3, due to its universal existence for the studied problems. For the problems we studied, it generally holds that finding a solution such that no individual has reason to reject is easy (by combining the individual solutions); however, by adding a constraint on the global quality of a solution in Chapter 4, determining the existence of an individually rational solution (and therefore finding such a solution) becomes intractable. We were able to achieve

some tractability results by looking at parameterized complexity in this setting: Among others, the target value of the quality of a solution and the maximum processing time of a job (especially in combination with other parameters) turned out to be powerful.

Generally, deciding whether a core-stable solution exists is intractable. For some of the problems we studied, however, a core-stable solution always exists, which makes this decision trivial. Namely, in Chapter 3, the partitioned minimum spanning tree game always admits a core-stable solution, inherited from its unpartitioned variant. In Chapter 5, the λ_1-NTU-MW game always admits a core-stable solution for $\lambda_1 \in \{\mathrm{AV}, \mathrm{SAV}, \mathrm{CC}, \mathrm{PAV}\}$[1], and the λ_2-TU-MW game for every totally separable voting rule λ_2 always admits a core-stable solution. In Chapter 3, we additionally looked at the parameterized complexity with respect to the number of agents and maximum size of an agent. While the maximum size of an agent did not lead to tractability, the number of agents led to FPT results for PCOGs, where the underlying optimization problem is tractable. This follows from the fact that we could explicitly compute and list all coalition constraints of the LP-formulations in FPT-time.

For core stability, we additionally considered the verification problem, which turned out to be intractable for all our studied problems. Again, for PCOGs where the underlying optimization problem is tractable, the verification problem is FPT with respect to the number of agents, with the same reasoning as for the existence case.

In total, the first challenge turned out to be intractable in most cases, with a few islands of tractability persisting.

In Chapter 6, we tackled the second challenge in the context of multi-winner voting. Here, we were able to derive a polynomial-time algorithm for computing a shortest reconfiguration path for the voting rules AV and SAV. Such a reconfiguration path is guaranteed to exist, as long as the target committee satisfies the score-bound. For the voting rules CC and PAV, the picture looks different: Not only is a reconfiguration path not guaranteed to exist, but determining the existence is PSPACE-complete and therefore intractable. In terms of parameterized complexity, two parameters proved powerful: the number of alternatives and the number of voters. The number of alternatives let us enumerate all possible committees. The number of voters yielded an FPT result for CC on its own, and an FPT result for PAV when combined with any other parameter.

In total, the tractability of the second challenge mirrors that of the underlying voting rules: It is tractable under AV and SAV and intractable under CC and PAV.

We want to highlight one technique we used in this thesis: the ellipsoid method. We applied the ellipsoid method in Chapters 3 and 5[2] to show membership in some complexity classes, such as Θ_2^{P} and coNP. To the best of our knowledge, the ellipsoid method has not been used in this way before. Perhaps future work can apply it to other problems, e.g., in the context of other TU games.

[1] For CC this is our own result; for AV, SAV, and PAV it follows from a result of Becker et al. [BGP26] together with our equivalence between the induced cores.

[2] E.g., Theorem 3.21.

Finally, for each of the chapters, we want to highlight some interesting open questions and possible future research directions.

In Chapter 3, we gave a complete complexity overview, so we turn directly to future directions. The PCOG model can be applied to a wide array of other problems studied in the COG framework. It could also be applied to our MW game setting.

In Chapter 4, the parameterized complexity of MOS (for both $\mathsf{C}_{\mathsf{max}}$ and ΣC) with respect to the maximum processing time $\mathsf{p}_{\mathsf{max}}$ is an intriguing open question. While we have an XP algorithm for $\mathsf{C}_{\mathsf{max}}$, we do not have a corresponding hardness result, raising the question of whether the problem is FPT. For ΣC, we have no result of any kind. Studying core stability in the MOS setting would be a natural extension of our model. In the context of core stability, removing the target objective would be sensible, as we do not know whether a core-stable solution is guaranteed to exist.

In Chapter 5, the exact complexities of multiple problems remain open: TU-core existence for CC and PAV, and TU-core verification for PAV. Studying MW games in a non-approval setting and extending the framework to participatory budgeting—where the alternatives in the committee have an associated cost, and there is a bound on the total cost of a committee—are promising future research avenues.

In Chapter 6, the exact parameterized complexity of PAV-MR with respect to the number of voters is interesting. CC-MR is FPT with respect to this parameter. While we have an XP-algorithm for PAV-MR, we do not have a matching lower bound, and thus it remains open whether PAV-MR is FPT with respect to the number of voters. As for future research directions, there are multiple problems that have not been studied in the reconfiguration framework, such as hedonic games or judgment aggregation.

Taken together, our results paint a consistent picture: Deciding how agents should cooperate, or how a shared decision should evolve, is hard far more often than not. However, structural assumptions or well-chosen parameters often led to tractability, suggesting that the barrier to computing cooperation lies less in the goal itself than in the underlying structure of the instance.

Overview of Generative AI Tools Used

ChatGPT GPT-5.3, Claude Sonnet and Opus 4.6/4.7/4.8, and Grammarly were used for proofreading assistance and to help with wording.

Übersicht verwendeter Hilfsmittel

ChatGPT GPT-5.3, Claude Sonnet und Opus 4.6/4.7/4.8 und Grammarly wurden zur Hilfestellung beim Korrekturlesen und als Formulierungshilfe verwendet.

List of Figures

1.1 An instance of the lecture-hall scenario, drawn as a graph. Each vertex represents an institute building, and an edge connects two buildings that are close enough to share a lecture hall. The three institutes are distinguished by color, and the differently shaded region of each color encloses the buildings owned by the corresponding institute. The two circled buildings r_3 and b_3 form a minimum dominating set: placing a lecture hall at r_3 and at b_3 puts every building within reach of a lecture hall. Both halls serve buildings of all three institutes, so two shared halls suffice, whereas four halls would be needed if each institute covered its own buildings alone. 2

2.1 Examples of the graph classes introduced in this subsection: (a) the complete bipartite graph $K_{2,3}$, (b) a forest, (c) a tree, (d) a star, (e) a path, and (f) a cycle. The circles are vertices, and the lines connecting the vertices are edges. 11

2.2 A graph $G = (V, E)$ with the 6 vertices $v_1, \ldots, v_6$ and the 8 edges $e_1, \ldots, e_8$. 12

2.3 The feasible region (shaded) of the system in Example 2.34. Each dashed line is the boundary of one of the three inequalities (2.7)–(2.9); their common side is the shaded triangle. As an LP it is non-empty—for example, $(\frac{1}{2}, \frac{1}{2})$ is feasible—but it contains no integral solution, so as an ILP it is infeasible. 22

2.4 The graph $G = (V, E)$ used in Examples 2.60 and 2.66. The vertices $\{v_1, v_2, v_5, v_6\}$ form a clique of size 4, while v_3 and v_4 form an isolated edge. 35

2.5 The bipartite graph $G = (U, E)$ used in Example 2.64, with parts $\{a_1, a_2, a_3\}$ and $\{b_1, b_2, b_3\}$. The four edges among $\{a_1, a_2\}$ and $\{b_1, b_2\}$ form a complete bipartite subgraph $K_{2,2}$. 38

2.6 Graph G and a dominating-set reconfiguration sequence 39

3.1 Illustration for the proof of Theorem 3.16 when $c_{\max} = 7$, showing the gadget for one variable x_1 occurring in the clause $C = (x_1 \vee \ell_j \vee \ell_k)$. The positive and negative literal vertices $v_{x_1}, v_{\overline{x}_1}$ and the clause vertices $v_C^{x_1}, v_C^{\ell_j}, v_C^{\ell_k}$ are joined through the chain vertices v_1^1, v_1^2, v_1^3; the patterns around the edges indicate which of the seven agents owns each edge set. Also visible are the chain vertices corresponding to x_2 and x_{n_1}. 58

3.2 Illustration for the proof of Theorem 3.23. The connector vertex v' is shared by agents 1 and 2 (edges $\{v', u_1\}$ and $\{v', u_2\}$), the edge $\{u_1, u_2\}$ belongs to agent 3, and the dashed green ellipse marks agent 4's subgraph G. The lined red pattern indicates agent 1, the blue grid pattern agent 2, solid yellow agent 3, and the green dashed outline agent 4. 66

4.1 Schedules for the instance in Example 4.1. Each job is represented by a rectangle whose length equals its processing time; jobs in the same row run on the same machine, and time runs from left to right, so a job drawn further left is processed earlier. The completion time of a job is the time at which it finishes, i.e., the right edge of its rectangle. O_1's jobs are blue and have a dashed boundary. O_2's jobs are red and have a solid boundary. 104
4.2 Pairing of the instances used to produce the gadgets 108
4.3 Representation of the jobs and machines of organization $O_{1+5(i-1)}$. 109
4.4 Representation of the jobs and machines of organization $O_{2+5(i-1)}$. 110
4.5 Representation of the jobs and machines of organization $O_{3+5(i-1)}$. 111
4.6 Representation of the jobs and machines of organization $O_{4+5(i-1)}$. 111
4.7 Representation of the jobs and machines of organization $O_{5+5(i-1)}$. 112
4.8 Structure of an existing individually rational schedule with makespan τ, relative to the i-th gadget, if one exists. The space left blank in the bottom machines corresponds to the time in which jobs from the sets $\mathcal{X}_1^i$,$\mathcal{X}_2^i$, and $\mathcal{E}_3^i$ have to be scheduled. Jobs in black are owned by the global equalizing organization. 117
4.9 Possible local schedule for the first organization. 122
4.10 Possible local schedule for the second organization. 123
4.11 Representation of a difference in end of phase b between two machines z_1 and z_2. Jobs of phases b and before are represented in blue with dashed lines, jobs from J_2 are represented in red with solid lines. 127

6.1 Reconfiguration graphs for Example 6.5 under CC and PAV. Left: $\delta_s = 0$ under both CC and PAV. Right: $\delta_s = 1$ under CC or $\delta_s = \frac{1}{2}$ under PAV. 206
6.2 Illustration for the proof of Lemma 6.16, depicting how the two committees W and W' overlap and the blocks A, B, W_1, B_1, and X. Left: Case 1; right: Case 2. 219
6.3 Construction from Proposition 6.19 for $k_{DS} = 1$ and $\hat{G} = K_3$, the complete graph with three vertices, i.e., a triangle. 224

List of Tables

3.1 Overview of the (parameterized) complexity of Core-Stability Verification and Core-Stability Existence for four Partitioned Combinatorial Optimization Games. Results in green are tractable, results in red are intractable ("-c" = complete, "-h" = hard). "P*" means that the corresponding game always admits a core-stable allocation, and finding one can be done in polynomial time. Results obtained in prior work are cited in the same cell; all other results are obtained in this thesis. References [T x], [P x], and [O x] point to Theorem, Proposition, and Observation x of this thesis, respectively. "–" denotes a case with no separate result. 45

4.1 (Parameterized) complexity of the makespan and the sum-of-completion-times variants of MOS. Results in green are tractable, results in red are intractable ("-c" = complete, "-h" = hard); within a cell, two comma-separated entries give a lower and an upper bound, and "?" marks an open case. "DP-h" means the problem remains DP-hard even if the value of the corresponding parameter is a constant. "NP-c" for the parameter combination $n_{\mathsf{max}} + \mathsf{m}_{\mathsf{max}}$ means that the decision variant is contained in NP when either of the parameters is a constant and it remains NP-hard even if both parameters have values bounded by a constant. Results obtained in prior work are cited in the same cell: The NP-completeness for $n_{\mathsf{max}} + \mathsf{m}_{\mathsf{max}}$ follows rather directly from Cohen et al. [CCTW11]. All other results are obtained in this thesis. References [T x], [P x], and [C x] point to Theorem, Proposition, and Corollary x of this thesis. Note that all omitted two-parameter combinations either have one parameter subsumed by the other, or have the result follow directly from another. 102

5.1 Overview of structural and computational results. Each cell reports whether the core is guaranteed to be non-empty and the complexity of determining the core-nonemptiness and verifying whether a given utility function lies in the core, respectively. Constructive polynomial-time algorithms exist for TU-AV, TU-SAV (Algorithm 5.1), and for NTU constructive polynomial-time algorithms exist for all our studied rules (Becker et al. [BGP26] and Algorithm 5.2, via the equivalence with JR). Results in green are tractable (P = constructive in polynomial time), results in red are intractable ("-c" = complete, "-h" = hard). The upper bounds DP, Θ_2^{P}, and Δ_2^{P} follow from Theorem 5.24. References [T x] and [P x] point to Theorem and Proposition x of this thesis. 149

6.1 (Parameterized) complexity results obtained in this chapter. We omit the results for AV and SAV as reconfiguration under both rules is polynomial-time solvable (see Theorem 6.7). Both PSPACE-hardness and W[1]-hardness hold even if each voter approves of at most two alternatives, i.e., $\mathsf{b} = 2$. Except for the NP-hardness for constant $\mathsf{b} + \ell$, all hardness results hold even if $(\delta_{\mathrm{c}}, \delta_{\mathrm{s}}) = (1, 0)$. Results in green are tractable, results in red are intractable ("-c" = complete, "-h" = hard); within a cell, two comma-separated entries give a lower and an upper bound, "?" marks an open case, and "–" a missing reference. References [T x] and [P x] point to Theorem and Proposition x of this chapter. 202

Glossary

allocation A vector assigning each player a non-negative utility in a TU game. 28

approval preference profile A triple $(\mathcal{C}, \mathcal{V}, \mathcal{R})$ of alternatives, voters, and the approval ballots specifying which alternatives each voter approves. 32

blocking coalition A coalition that can improve all of its members' utilities relative to a proposed outcome. 27

certificate A witness for an instance of a problem being a yes-/no-instance. 14

characteristic function The function mapping every coalition to a non-negative value in a TU game. 28

combinatorial optimization game (COG) A cooperative game defined on a combinatorial structure (e.g., a graph) in which each agent owns a single unit of the structure and the value of a coalition is determined by the optimal solution of a combinatorial optimization problem restricted to the substructure the coalition owns. 41

committee A subset of alternatives. 32

coNP The class of decision problems whose no-instances admit a polynomial-size certificate that is verifiable in polynomial time. 15

core (core stability) The set of feasible outcomes/preimputations that no coalition can block. An outcome in the core is called core-stable. 28

dominating set game A combinatorial optimization game in which agents correspond to vertices and a coalition's value is the minimum dominating set size of its induced subgraph. 41

DP The class of problems that can be expressed as the intersection of an NP and a coNP problem. 16

ellipsoid method A polynomial-time algorithm for linear programming, usable even when the constraints are not explicitly given. 21

fixed-parameter tractable (FPT) A parameterized problem solvable in time $f(k) \cdot |I|^{O(1)}$ for a computable function f of the parameter k. The class of fixed-parameter tractable problems is FPT. 18

individual rationality (IR) The requirement that no player is worse off in the grand coalition than acting alone. 27

integer linear program (ILP) A linear program in which the variables are required to take integer values. 20

linear program (LP) A problem consisting of linear constraints over real variables aiming to optimize an objective. 21

local schedule A schedule in which every job of an organization runs only on machines owned by that organization. An optimal local schedule minimizes the makespan (resp. the sum of completion times). 103

makespan The completion time of the last job to finish under a schedule. 31

maximization game A cooperative game in which players prefer larger values. 27

maximum matching game A combinatorial optimization game in which agents correspond to vertices and a coalition's value is the maximum matching size of its induced subgraph. 41

minimization game A cooperative game in which players prefer smaller costs. 27

minimum cost spanning tree game A combinatorial optimization game in which agents correspond to vertices and a coalition's value is the weight of a minimum cost spanning tree including a predetermined source vertex. 41

multi-organizational scheduling (MOS) The scheduling setting studied in this thesis in which several organizations, each owning its own machines and jobs, cooperate on a shared schedule subject to individual rationality. 103

multi-winner approval voting The setting of selecting a fixed-size committee of alternatives from an approval preference profile. 32

multi-winner reconfiguration The problem of transforming one size-k committee into another via a sequence of intermediate size-k committees that stay close to one another and satisfy a score-bound. 199

non-transferable utility (NTU) game A cooperative game (N, μ) assigning each coalition a set of feasible payoff vectors. 27

NP The class of decision problems whose yes-instances admit a polynomial-size certificate that is verifiable in polynomial time. 14

NP-hard A problem to which every problem in NP can be reduced in polynomial time; NP-complete if it is also in NP. 15

para-NP-hard A parameterized problem that is NP-hard already for some constant value of the parameter. 19

parameterized problem A decision problem together with a parameter. 18

partitioned combinatorial optimization game (PCOG) A cooperative game defined on a combinatorial structure (e.g., a graph) in which each agent owns a part of the structure and the value of a coalition is determined by the optimal solution of a combinatorial optimization problem restricted to the substructure that the coalition owns. PCOGs generalize COGs. 42

Partitioned Dominating Set Game The partitioned variant of the dominating set game. 48

Partitioned Matching Game The partitioned variant of the maximum matching game. 50

Partitioned Spanning Tree Game The partitioned variant of the minimum cost spanning tree game. 49

Partitioned Vertex Cover Game The partitioned variant of the vertex cover game. 47

polynomial-time many-one reduction An algorithm that transforms an instance of one problem into an instance of another problem, while preserving yes-/no-instances. 14

preimputation An allocation that distributes exactly the grand coalition's value among the agents in a TU game. 29

PSPACE The class of problems that can be solved using polynomial space. 17

reconfiguration The transformation of one solution into another via small, feasible incremental changes. 199

reconfiguration path A sequence of feasible solutions connecting two given solutions in which consecutive solutions are close under some criteria. In the case of multi-winner reconfiguration, the solutions are committees. 204

reconfiguration problem A problem asking whether one feasible solution can be transformed into another through a sequence of feasible, pairwise-adjacent solutions. 39

separation oracle An algorithm that, given a point, either confirms it lies in the solution or returns a violated constraint. 25

sum of completion times The total of all jobs' completion times under a schedule. 31

transferable utility (TU) game A cooperative game (N, ν) with a characteristic function assigning each coalition a single real value that its members may freely divide. 28

vertex cover game A combinatorial optimization game in which agents correspond to edges and a coalition's value is the minimum vertex cover size of its induced subgraph. 41

W-hierarchy The hierarchy of parameterized complexity classes W[1], W[2], ... between FPT and XP. 19

XP The class of parameterized problems solvable in time $|I|^{f(k)}$ for a computable function f and parameter k. 18

Bibliography

[AAB+17] Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Timothy M. Chan, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Penny Haxell, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T. Wilkinson. How to morph planar graph drawings. *SIAM Journal on Computing*, 46(2):824–852, 2017.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ABC+17] Haris Aziz, Markus Brill, Vincent Conitzer, Edith Elkind, Rupert Freeman, and Toby Walsh. Justified representation in approval-based committee voting. *Social Choice and Welfare*, 48(2):461–485, 2017.

[ACH+26] Clemens Anzinger, Jiehua Chen, Christian Hatschka, Manuel Sorge, and Alexander Temper. How hard is it to explain preferences using few Boolean attributes? In *Proceedings of the 40th AAAI Conference on Artificial Intelligence (AAAI 2026)*, pages 16639–16646, 2026.

[AEOP02] Ravindra K. Ahuja, Özlem Ergun, James B. Orlin, and Abraham P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002.

[AGG+15] Haris Aziz, Serge Gaspers, Joachim Gudmundsson, Simon Mackenzie, Nicholas Mattei, and Toby Walsh. Computational aspects of multi-winner approval voting. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, pages 107–115, 2015.

[AMP15] Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discrete & Computational Geometry*, 54(2):368–389, 2015.

[AR12] Itai Ashlagi and Alvin E. Roth. New challenges in multihospital kidney exchange. *American Economic Review*, 102(3):354–359, 2012.

[AR14] Itai Ashlagi and Alvin E. Roth. Free riding and participation in large scale, multi-hospital kidney exchange. *Theoretical Economics*, 9(3):817–863, 2014.

[AS99] Alexander A. Ageev and Maxim Sviridenko. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In *Proceedings of the 7th International Conference on Integer Programming and Combinatorial Optimization (IPCO 1999)*, volume 1610, pages 17–30, 1999.

[BBK+25] Márton Benedek, Péter Biró, Walter Kern, Dömötör Pálvölgyi, and Daniël Paulusma. Partitioned matching games for international kidney exchange. *Mathematical Programming*, 214(1):723–758, 2025.

[BBPY24] Márton Benedek, Péter Biró, Daniël Paulusma, and Xin Ye. Computing balanced solutions for large international kidney exchange schemes. *Autonomous Agents and Multi-Agent Systems*, 38(1):18, 2024.

[BCK+26] Robert Bredereck, Jiehua Chen, Dušan Knop, Junjie Luo, and Rolf Niedermeier. Adapting stable matchings to evolving preferences. *Games and Economic Behavior*, 157:322–350, 2026.

[BFN+20] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, Piotr Skowron, and Nimrod Talmon. Mixed integer programming with convex/concave constraints: fixed-parameter tractability and applications to multicovering and voting. *Theoretical Computer Science*, 814:86–105, 2020.

[BG93] Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, 22(3):560–572, 1993.

[BGP+24] Markus Brill, Paul Gölz, Dominik Peters, Ulrike Schmidt-Kraepelin, and Kai Wilker. Approval-based apportionment. *Mathematical Programming*, 203(1):77–105, 2024.

[BGP26] Patrick Becker, Matthias Greger, and Dominik Peters. Existence of the core in approval-based committee elections. arXiv:2609.11912, 2026.

[BGS26] Hans L. Bodlaender, Carla Groenland, and Céline M. F. Swennenhuis. Parameterized complexities of dominating and independent set reconfiguration. *Algorithmica*, 88(3):39, 2026.

[BHH01] Peter Borm, Herbert Hamers, and Ruud Hendrickx. Operations research games: a survey. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 9:139–199, 2001.

[Bir76] C. G. Bird. On cost allocation for a spanning tree: a game theoretic approach. *Networks*, 6(4):335–350, 1976.

[BKN20] Robert Bredereck, Andrzej Kaczmarczyk, and Rolf Niedermeier. Electing successive committees: Complexity and algorithms. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, pages 1846–1853, 2020.

[BKPP19] Péter Biró, Walter Kern, Dömötör Pálvölgyi, and Daniël Paulusma. Generalized matching games for international kidney exchange. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2019)*, pages 413–421, 2019.

[BKPW18] Péter Biró, Walter Kern, Daniël Paulusma, and Péter Wojuteczky. The stable fixtures problem with payments. *Games and Economic Behavior*, 108:245–268, 2018.

[BKS03] Piotr Berman, Marek Karpinski, and Alexander D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. Technical Report TR03-049, Electronic Colloquium on Computational Complexity, 2003.

[BKT98] Harry Buhrman, Jim Kadin, and Thomas Thierauf. Functions computable with nonadaptive queries to NP. *Theory of Computing Systems*, 31(1):77–92, 1998.

[BM14] Péter Biró and Eric McDermid. Matching with sizes (or scheduling with processing set restrictions). *Discrete Applied Mathematics*, 164(1):61–67, 2014.

[BN21] Niclas Boehmer and Rolf Niedermeier. Broadening the research agenda for computational social choice: Multiple preference profiles and multiple solutions. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, pages 1–5, 2021.

[Bon63] Olga N Bondareva. Some applications of linear programming methods to the theory of cooperative games. *Problemy Kibernet*, 10:119, 1963.

[Bon13] Paul Bonsma. The complexity of rerouting shortest paths. *Theoretical Computer Science*, 510:1–12, 2013.

[Bru04] Peter Brucker. *Scheduling algorithms (4. ed.)*. Springer, 2004.

[BSU13] Nadja Betzler, Arkadii Slinko, and Johannes Uhlmann. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, 47:475–519, 2013.

[BT96] Harry Buhrman and Thomas Thierauf. The complexity of generating and checking proofs of membership. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1996)*, pages 75–86, 1996.

[BTC+26] Ratip Emin Berker, Emanuel Tewolde, Vincent Conitzer, Mingyu Guo, Marijn Heule, and Lirong Xia. On the edge of core (non-)emptiness: an automated reasoning approach to approval-based multi-winner voting. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence (AAAI 2026)*, pages 16673–16681, 2026.

[CCR14] Johanne Cohen, Daniel Cordeiro, and Pedro Luis F Raphael. Energy-aware multi-organization scheduling problem. In *Proceedings of the 20th International Conference on Parallel and Distributed Computing (Euro-Par 2014)*, pages 186–197, 2014.

[CCTW11] Johanne Cohen, Daniel Cordeiro, Denis Trystram, and Frédéric Wagner. Multi-organization scheduling approximation algorithms. *Concurrency and Computation: Practice and Experience*, 23(17):2220–2234, 2011.

[CDH25] Jiehua Chen, Martin Durand, and Christian Hatschka. Multi-organizational scheduling: Individual rationality, optimality, and complexity. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI 2025)*, pages 3780–3787, 2025.

[CDMT11] Daniel Cordeiro, Pierre-François Dutot, Grégory Mounié, and Denis Trystram. Tight analysis of relaxed multi-organization scheduling algorithms. In *Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2011)*, pages 1177–1186, 2011.

[CDR03] Robert Connelly, Erik D. Demaine, and Günter Rote. Blowing up polygonal linkages. *Discrete & Computational Geometry*, 30(2):205–239, 2003.

[CEW11] Georgios Chalkiadakis, Edith Elkind, and Michael J. Wooldridge. *Computational Aspects of Cooperative Game Theory*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[CFK+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[CGL+13] Anirudh Chakravorty, Neelima Gupta, Neha Lawaria, Pankaj Kumar, and Yogish Sabharwal. Algorithms for the relaxed multiple-organization multiple-machine scheduling problem. In *Proceedings of the 20th Annual International Conference on High Performance Computing*, pages 30–38, 2013.

[CHS23] Jiehua Chen, Christian Hatschka, and Sofia Simola. Efficient algorithms for Monroe and CC rules in multi-winner elections with (nearly) structured preferences. In *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI 2023)*, volume 372, pages 397–404, 2023.

[CHS24] Jiehua Chen, Christian Hatschka, and Sofia Simola. Multi-winner reconfiguration. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems (NeurIPS 2024)*, pages 85899–85931, 2024.

[CHS25] Jiehua Chen, Christian Hatschka, and Sofia Simola. Partitioned combinatorial optimization games. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*, volume 413, pages 1429–1436, 2025.

[CHS26] Jiehua Chen, Christian Hatschka, and Sofia Simola. Computational social choice: Parameterized complexity and challenges. *Computer Science Review*, 61:100835, 2026.

[Chv83] V. Chvátal. *Linear Programming*. Series of books in the mathematical sciences. W. H. Freeman, 1983.

[CJMW20] Yu Cheng, Zhihao Jiang, Kamesh Munagala, and Kangning Wang. Group fairness in committee selection. *ACM Transactions on Economics and Computation*, 8(4):23:1–23:18, 2020.

[CK73] A Claus and DJ Kleitman. Cost allocation for a spanning tree. *Networks*, 3(4):289–304, 1973.

[CLR+25] Moses Charikar, Alexandra Lassota, Prasanna Ramakrishnan, Adrian Vetta, and Kangning Wang. Six candidates suffice to win a voter majority. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC 2025)*, pages 1590–1601, 2025.

[CvdHJ11] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67(1):69–82, 2011.

[CZC14] Michele Conforti, Giacomo Zambelli, and Gérard P Cornuéjols. *Integer Programming*. Graduate Texts in Mathematics. Springer International Publishing, 2014.

[Den09] Xiaotie Deng. Combinatorial optimization games. In *Encyclopedia of Optimization, Second Edition*, pages 387–391. Springer, 2009.

[DF95a] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[DF95b] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: on completeness for W[1]. *Theoretical Computer Science*, 141(1&2):109–131, 1995.

[DF13] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[DFPS26] Chris Dong, Fabian Frank, Jannik Peters, and Warut Suksompong. Reconfiguring proportional committees. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence (AAAI 2026)*, pages 16829–16837, 2026.

[DG80] J. H. Drèze and J. Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica*, 48(4):987–1003, 1980.

[DIN99] Xiaotie Deng, Toshihide Ibaraki, and Hiroshi Nagamochi. Algorithmic aspects of the core of combinatorial optimization games. *Mathematics of Operations Research*, 24(3):751–766, 1999.

[DP21] Martin Durand and Fanny Pascual. Efficiency and equity in the multi organization scheduling problem. *Theoretical Computer Science*, 864:103–117, 2021.

[EdLW24] Paul H. Edelman, Martin Van der Linden, and John A. Weymark. The core of a transferable utility game as the solution to a public good market demand problem. *Mathematical Programming*, 203(1):687–702, 2024.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[EFH20] Arantza Estévez-Fernández and Herbert Hamers. Chinese postman games with multi-located players. *European Journal of Operational Research*, 285(2):458–469, 2020.

[EGGW09] Edith Elkind, Leslie Ann Goldberg, Paul W. Goldberg, and Michael J. Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[FGM16] Brandon Fain, Ashish Goel, and Kamesh Munagala. The core of the participatory budgeting problem. In *Proceedings of the 12th International Conference on Web and Internet Economics (WINE 2016)*, volume 10123, pages 384–399, 2016.

[FJK+25] Piotr Faliszewski, Lukasz Janeczko, Andrzej Kaczmarczyk, Grzegorz Lisowski, and Grzegorz Pierczynski. Project submission games in participatory budgeting. arXiv:2508.09741, 2025.

[FK93] Ulrich Faigle and Walter Kern. On some approximately balanced combinatorial cooperative games. *ZOR – Methods and Models of Operations Research*, 38(2):141–152, 1993.

[FK05] Qizhi Fang and Hye Kyung Kim. A note on balancedness of dominating set games. *Journal of Combinatorial Optimization*, 10(4):303–310, 2005.

[FKFH97] Ulrich Faigle, Walter Kern, Sándor P Fekete, and Winfried Hochstättler. On the complexity of testing membership in the core of min-cost spanning tree games. *International Journal of Game Theory*, 26:361–366, 1997.

[FKZ08] Qizhi Fang, Liang Kong, and Jia Zhao. Core stability of vertex cover games. *Internet Mathematics*, 5(4):383–394, 2008.

[FMS18] Brandon Fain, Kamesh Munagala, and Nisarg Shah. Fair allocation of indivisible public goods. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC 2018)*, pages 575–592, 2018.

[FT87] András Frank and Éva Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[GH81] Daniel Granot and Gur Huberman. Minimum cost spanning tree games. *Mathematical Programming*, 21(1):1–18, 1981.

[GJ75] M. R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GJS76] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[GKMP09] Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou. The connectivity of Boolean satisfiability: computational and structural dichotomies. *SIAM Journal on Computing*, 38(6):2330–2355, 2009.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[GMP12] Laurent Gourvès, Jérôme Monnot, and Fanny Pascual. Cooperation in multiorganization matching. *Algorithmic Operations Research*, 7(2):111–124, 2012.

[Gon85] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[GS62] David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[GS18] Sevag Gharibian and Jamie Sikora. Ground state connectivity of local Hamiltonians. *ACM Transactions on Computation Theory*, 10(2):8:1–8:28, 2018.

[GSV25] Matthew Gerstbrein, Laura Sanità, and Lucy Verberk. Stabilization of capacitated matching games. *Mathematical Programming*, 210(1):313–334, 2025.

[Gur26] Gurobi Optimization, LLC. Gurobi Optimization — gurobi.com. `www.gurobi.com`, 2026. [Accessed 17-01-2026].

[Gus20] Vasily V. Gusev. The vertex cover game: Application to transport networks. *Omega*, 97:102102, 2020.

[HBvdLT99] Herbert Hamers, Peter Borm, Robert van de Leensel, and Stef Tijs. Cost allocation in the Chinese postman problem. *European Journal of Operational Research*, 118(1):153–163, 1999.

[HIM+16] Arash Haddadan, Takehiro Ito, Amer E. Mouawad, Naomi Nishimura, Hirotaka Ono, Akira Suzuki, and Youcef Tebbal. The complexity of dominating set reconfiguration. *Theoretical Computer Science*, 651:37–49, 2016.

[HKMS24] Adrian Haret, Sophie Klumper, Jan Maly, and Guido Schäfer. Committees and equilibria: Multiwinner approval voting through the lens of budgeting games. In *Proceedings of the 25th ACM Conference on Economics and Computation (EC 2024)*, pages 51–70, 2024.

[HN16] Ashkan Hafezalkotob and Fateme Naseri. Cooperative network flow problem with pricing decisions and allocation of benefits: a game theory approach. *Journal of Industrial and Systems Engineering*, 9:73–87, 2016. Special issue on supply chain.

[HSV05] Edith Hemaspaandra, Holger Spakowski, and Jörg Vogel. The complexity of Kemeny elections. *Theoretical Computer Science*, 349(3):382–391, 2005.

[IBM26] IBM ILOG. IBM ILOG CPLEX Optimization Studio — ibm.com. `https://www.ibm.com/products/ilog-cplex-optimization-studio`, 2026. [Accessed 17-01-2026].

[IDH+11] Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011.

[IIK+25] Takehiro Ito, Yuni Iwamasa, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, Yuta Nozaki, Yoshio Okamoto, and Kenta Ozeki. Reforming an envy-free matching. *Algorithmica*, 87(4):594–620, 2025.

[IKO+14] Takehiro Ito, Marcin Kaminski, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka. On the parameterized complexity for token jumping on graphs. In *Proceedings of the 11th International Conference on Theory and Applications of Models of Computation (TAMC 2014)*, pages 341–351, 2014.

[IKSY24] Ayumi Igarashi, Naoyuki Kamiyama, Warut Suksompong, and Sheung Man Yuen. Reachability of fair allocations via sequential exchanges. *Algorithmica*, 86(12):3653–3683, 2024.

[JKMS13] Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.

[JMW20] Zhihao Jiang, Kamesh Munagala, and Kangning Wang. Approximately stable committee selection. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020)*, pages 463–472, 2020.

[Joh90] David S. Johnson. A catalog of complexity classes. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 67–161. Elsevier and MIT Press, 1990.

[JS79] William W. Johnson and William E. Story. Notes on the "15" puzzle. *American Journal of Mathematics*, 2(4):397–404, 1879.

[JYW17] Chengquan Ju, Shuhan Yao, and Peng Wang. Resilient post-disaster system reconfiguration for multiple energy service restoration. In *Proceedings of the 1st IEEE Conference on Energy Internet and Energy System Integration (EI^2 2017)*, pages 1–6, 2017.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of the Symposium on the Complexity of Computer Computations (CCC 1972)*, pages 85–103, 1972.

[Kha79] Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[KKM20] Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. *Mathematical Programming*, 184:1–34, 2020.

[Kre88] Mark W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[KVPS21] Xenia Klimentova, Ana Viana, João Pedro Pedroso, and Nicolau Santos. Fairness models for multi-agent kidney exchange programmes. *Omega*, 102:1–14, 2021.

[LB11] Tyler Lu and Craig Boutilier. Budgeted social choice: from consensus to personalized decision making. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 280–286, 2011.

[Len83] Hendrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[Lin18] Bingkai Lin. The parameterized complexity of the k-biclique problem. *Journal of the ACM*, 65(5):34:1–34:23, 2018.

[LL08] Joseph Y.-T. Leung and Chung-Lun Li. Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, 116(2):251–262, 2008.

[LL16] Joseph Y.-T. Leung and Chung-Lun Li. Scheduling with processing set restrictions: A literature update. *International Journal of Production Economics*, 175:1–11, 2016.

[LM17] Thomas Lukasiewicz and Enrico Malizia. A novel characterization of the complexity class Θ_k^{p} based on counting and comparison. *Theoretical Computer Science*, 694:21–33, 2017.

[LNB16] Phuoc Hoang Le, Tri-Dung Nguyen, and Tolga Bektas. Generalized minimum spanning tree games. *EURO Journal on Computational Optimization*, 4(2):167–188, 2016.

[LS23] Martin Lackner and Piotr Skowron. *Multi-Winner Voting with Approval Preferences–Artificial Intelligence, Multiagent Systems, and Cognitive Robotics*. Springer Briefs in Intelligent Systems. Springer, 2023.

[Mal25] Jan Maly. The core in participatory budgeting can be empty. *Economics Letters*, 254:112472, 2025.

[MNR+17] Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, Narges Simjour, and Akira Suzuki. On the parameterized complexity of reconfiguration problems. *Algorithmica*, 78(1):274–297, 2017.

[Mon95] Burt L Monroe. Fully proportional representation. *American Political Science Review*, 89(4):925–940, 1995.

[MSW22] Kamesh Munagala, Yiheng Shen, and Kangning Wang. Auditing for core stability in participatory budgeting. In *Proceedings of the 18th International Conference on Web and Internet Economics (WINE 2022)*, volume 13778, pages 292–310, 2022.

[MVB18] Matthias Mnich and René Van Bevern. Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research*, 100:254–261, 2018.

[MvVHN06] Silvia Miquel, Bas van Velzen, Herbert Hamers, and Henk Norde. Fixed tree games with multilocated players. *Networks*, 47(2):93–101, 2006.

[MW15] Matthias Mnich and Andreas Wiese. Scheduling and fixed-parameter tractability. *Mathematical Programming*, 154:533–562, 2015.

[Net24] Netflix. Don't Miss Your Chance to Watch These Movies and Shows Leaving in December. https://www.netflix.com/tudum/articles/whats-leaving-netflix, 2024. [Accessed 17-01-2024].

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[Nis18] Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.

[NYQ+25] Qingqin Nong, Jingyu Yao, Xin Qin, Suning Gong, and Qizhi Fang. Approximate cores of submodular cost set cover games. *Journal of Optimization Theory and Applications*, 207(3):53, 2025.

[OII09] Fukuhito Ooshita, Tomoko Izumi, and Taisuke Izumi. A generalized multi-organization scheduling on unrelated parallel machines. In *Proceedings of the 10th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2009)*, pages 26–33, 2009.

[OII12] Fukuhito Ooshita, Tomoko Izumi, and Taisuke Izumi. The price of multi-organization constraint in unrelated parallel machine scheduling. *Parallel Processing Letters*, 22(02):1250006, 2012.

[OPEG20] Svetlana Obraztsova, Maria Polukarov, Edith Elkind, and Marek Grzesiuk. Multiwinner candidacy games. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, pages 957–965, 2020.

[Pap81] Christos H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM*, 28(4):765–768, 1981.

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[PCT92] Jos AM Potters, Imma J Curiel, and Stef H Tijs. Traveling salesman games. *Mathematical Programming*, 53:199–211, 1992.

[Pet25] Dominik Peters. The core of approval-based committee elections with few seats. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI 2025)*, pages 4014–4022, 2025.

[Pin22] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing, 2022.

[Pri57] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[PRT09] Fanny Pascual, Krzysztof Rzadca, and Denis Trystram. Cooperation in multi-organization scheduling. *Concurrency and Computation: Practice and Experience*, 21(7):905–921, 2009.

[PRZ08] Ariel D Procaccia, Jeffrey S Rosenschein, and Aviv Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30:353–362, 2008.

[PS07] Bezalel Peleg and Peter Sudhölter. *Introduction to the theory of cooperative games*. Theory and Decision Library C. Springer, second edition, 2007.

[PS22] Grzegorz Pierczynski and Piotr Skowron. Core-stable committees under restricted domains. In *Proceedings of the 18th International Conference on Web and Internet Economics (WINE 2022)*, volume 13778, pages 311–329. Springer, 2022.

[PV24] Prime-Video. Prime Video: leaving soon. `https://www.amazon.co.uk/Expiring-Prime-Next-30-Days-Videos/s?rh=n:9791716031,p_n_ways_to_watch:7448660031`, 2024. [Accessed 17-01-2024].

[PW88] Christos H. Papadimitriou and David Wolfe. The complexity of facets resolved. *Journal of Computer and System Sciences*, 37(1):2–13, 1988.

[PY84] Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

[Rza07] Krzysztof Rzadca. Scheduling in multi-organization grids: Measuring the inefficiency of decentralization. In *Proceedings of the 7th International Conference on Parallel Processing and Applied Mathematics (PPAM 2007)*, volume 4967, pages 1048–1058, 2007.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[Sch99] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999.

[Sel94] Alan L. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Sciences*, 48(2):357–381, 1994.

[Sha67] Lloyd S. Shapley. On balanced sets and cores. *Naval Research Logistics Quarterly*, 14(4):453–460, 1967.

[Sha71] Lloyd S. Shapley. Core of convex games. *International Journal of Game Theory*, 1:11–26, 1971.

[SM73] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC 1973)*, pages 1–9, 1973.

[SN25] Haoyu Song and Thanh Nguyen. Approximate core of participatory budgeting via Lindahl equilibrium. Technical report, Purdue University, 2025. Available at `https://web.ics.purdue.edu/~nguye161/LEO.pdf`.

[SR14] Piotr Skowron and Krzysztof Rzadca. Fair share is not enough: measuring fairness in scheduling with cooperative game theory. In *Proceedings of the 10th International Conference on Parallel Processing and Applied Mathematics (PPAM 2013)*, pages 38–48, 2014.

[SSTT18] Baruch Schieber, Hadas Shachnai, Gal Tamir, and Tami Tamir. A theory and algorithms for combinatorial reoptimization. *Algorithmica*, 80(2):576–607, 2018.

[Sto76] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[SU13] Tayfun Sönmez and Utku M. Ünver. Market design for kidney exchange. In Nir Vulkan, Alvin E. Roth, and Zvika Neeman, editors, *The Handbook of Market Design*, pages 92–137. Oxford University Press, 2013.

[TSRB71] Constantine Toregas, Ralph Swain, Charles ReVelle, and Lyn Bergman. The location of emergency service facilities. *Operations Research*, 19(6):1363–1373, 1971.

[vdH13] Jan van den Heuvel. The complexity of change. In Simon R. Blackburn, Stefanie Gerke, and Mark Wildon, editors, *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013.

[vV04] Bas van Velzen. Dominating set games. *Operations Research Letters*, 32(6):565–573, 2004.

[Wag90] Klaus W. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

[XLLF25] Han Xiao, Yi-Han Li, Tian-Hang Lu, and Qi-Zhi Fang. Approximate core allocations for multiple partners matching games. *Journal of the Operations Research Society of China*, 2025.

[XWF21] Han Xiao, Yuanxi Wang, and Qizhi Fang. On the convexity of independent set games. *Discrete Applied Mathematics*, 291:271–276, 2021.

[YW23] Yongjie Yang and Jianxin Wang. Parameterized complexity of multi-winner determination: more effort towards fixed-parameter tractability. *Autonomous Agents and Multi-Agent Systems*, 37(2):28, 2023.