Fairness and Stability in Structured Environments

Ayumi Igarashi
St Catherine's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

March 2018

Abstract

In many social and economic situations, networks are the primary vehicle for strategic interactions among multiple players. On the one hand, networks connect individuals and represent how they interact with each other. On the other hand, networks describe relations between objects that are of interest to multiple decision-makers. The aims of the thesis are two-fold: first, to describe how the underlying network structures affect the existence of desirable outcomes of strategic interactions; and second, to discuss computational issues that arise when considering problems with connectivity constraints imposed by a network. In particular, we will consider two settings in which networks play a critical role: coalition formation and fair division restricted by networks. Specifically, we will study a model in which the space of feasible outcomes is restricted to the connected subgraphs of an underlying network. In essence, we show that acyclicity of a network is a necessary and sufficient condition for desirable outcomes to exist and to be efficiently computable.

Acknowledgements

I am most grateful to my supervisor, Prof. Edith Elkind. Over the course of my DPhil study, Edith has inspired me in many ways, has guided me patiently, and has encouraged my research continuously. I truly enjoyed collaborating with her and was very fortunate to have her as a supervisor. I would like to express my huge gratitude to Prof. Gerhard J Woeginger and Prof. Michael Wooldridge for their extremely helpful comments and insightful criticisms on the thesis. The wonderful viva experience with them will certainly make me more respectful and passionate about research. Also, I was very lucky to have worked with the cheerful colleagues in Oxford, including Dominik Peters, Paul Harrestein, Markus Brill, Martin Lackner, Robert Bredereck, Piotr Skowron, and Jiarui Gan. Especially, I want to thank Dominik, who has been a great colleague and co-author since the very beginning of my time at Oxford, for his friendship and constant support. I also thank Robert who has read parts of the thesis and made very valuable comments. A warm thank goes to Paul, who has been always helpful and constantly offering nice coffee. Further, I am indebted to many other researchers and collaborators across the world, such as Haris Aziz, Sylvain Bouveret, Katrina K. Cechlárová, Piotr Faliszewski, Angelo Fanelli, Rani Izsak, Jérôme Lang, Frédéric Meunier, Adèle Pass-Lanneau, Diederik Roijers, Jakub Sliwinski, and Yair Zick. Special thanks go to Yair Zick and Frédéric Meunier who generously invited me for a research visit, which proved very helpful and fruitful. I had the privilege of being a part of the COMSOC community. The friendly and pleasant nature of this community made the work with all the people I have mentioned delightful. I learned from the community that research is not just a mere activity of a single person but a continuing intellectual interaction with others. Last but not least, I have always appreciated the help and support that my family and friends have given me over the past 27 years, especially my parents who have been ignoring my faults and always by my side.

Preface

The results in this thesis are partially contained in conference publications [62, 63, 61, 26] and have been obtained in close collaboration with several co-authors.

First, I will discuss hedonic games with graph-restricted communication [62]. The research on this topic was stimulated by a discussion with my colleagues Dominik Peters and Paul Harrenstein, when they gave an internal talk about their work on hedonic games. I developed the model of hedonic games with graph-restricted communication together with my supervisor Edith Elkind, and obtained a number of existence and complexity results regarding stable outcomes in such a context. I presented the full paper at AAMAS 2016.

Second, I will discuss group activity selection problems on social networks [63, 61]. I proposed to work on this project to my supervisor in June 2016. Together with Dominik Peters, we obtained the NP-hardness result of computing a Nash stable outcome on a path. Robert Bredereck joined this project in September 2016; we came up with some proof ideas for W[1]-hardness results with respect to the number of activities for computing stable outcomes. The W[1]-hardness result with respect to the number of players is independently obtained by him, although I have contributed the writing of the proofs for core and individual stability. The corresponding results are published in AAAI 2017 and AAMAS 2017. Herein, I was leading the discussion, did most of the writing for the submissions, and contributed to many results such as fixed parameter tractability results.

Third, I will discuss learnability in hedonic graph games, which forms a part of the joint work with Yair Zick and Jakub Sliwinski. The collaboration was triggered when I visited Yair Zick at National University of Singapore. As an extension of their IJCAI paper [88], I proposed to work on the project. I contributed to this project by developing some basic proof ideas for the positive result for core stability on forests and proving the hardness of finding a path consistent with given samples. The paper is currently under submission.

Finally, I will discuss the fair division problem with graph connectivity constraints [26]. Together with Edith Elkind and Dominik Peters, we started the collaboration when Katrina K. Cechlárová introduced the model, which she developed with Sylvain Bouveret, at Budapest Workshop on Future Directions in Computational Social Choice. I have made major contributions to this submission, coming up with some fundamental ideas, and doing a substantial amount of writing.

Contents

Part I

Introduction and Basics

Chapter 1

Introduction

1.1 Introduction

Imagine yourself as a PhD student attending a conference banquet for the first time. You would like to socialize and possibly make new useful contacts. But the merest glance tells you that most people you see are all strangers and somehow your brain has picked out two figures who seem familiar. Who would you approach first? You may talk to those you know already and wish to get to know others later on through the mutual friends.

In many social and economic situations, the structure of networks is a key determinant in achieving desirable outcomes of strategic interactions. This thesis will aim to analyze how networks affect social and economic outcomes. Putting it crudely, there are two situations in which networks are critical. In one situation, the network structures describe how individuals interact and form groups; this includes the preceding example of a social network, emergence of nations, and alliances among political parties. In the other situation, the network can represent relations between actions or objectives that individuals may seek for. For example, the problem of how multiple countries divide a land falls into this category, where each piece of land corresponds to a node of the network and the adjacency relation corresponds to an edge.

Despite the existence of numerous important questions, this thesis will consider two problems constrained by networks. First, we will study coalition formation in which communication structure among players is represented by an undirected graph: a subset of players can form a coalition whenever they are connected in the underlying communication structure. Second, we consider how multiple participants can fairly divide the objects over a network. These two problems share the common feature that networks represent restrictions on individual decision making. The key questions across the thesis are the following:

- What is the impact of the structure of such network relationships determining the outcome of economic interactions?
- How computationally efficient is it to reach such desirable outcomes?

In answering these questions, we will primarily focus on *acyclicity* of a network, which in many contexts, allows desirable characterization as well as efficient calculation of outcomes.

Throughout, we assume that the relations defined by a network are exogenously given and static – it excludes the possibility of certain network formation. The framework this thesis will introduce is thus not the most general one, but we will demonstrate that it is one for which research is promising.

1.2 Thesis Outline

The thesis is organized into two parts. The first part, which contains Chapters 2, 3, and 4, discusses coalition formation problems where the network restricts possible grouping of individuals, whereas the second part is devoted to analyzing the fair division problem, where the network describes constraints on the division of goods.

1.2.1 Stable Coalition Formation in Structured Environments

Coalition formation arises everywhere in human activities: families are groups of people closely related by blood, clubs are social gatherings for people who share common interest, political parties are organizations of people with the same political purpose, and companies are large entities of people who unite in order to gain more profits. Individuals form a coalition in order to achieve the objectives that they cannot accomplish on their own.

Many important aspects of coalition formation can be studied using the formalism of *hedonic games* [16, 23]. In these games, each agent has preferences over all coalitions that she can be a part of, and an outcome is a partition of agents into coalitions. An important consideration in this context is *coalitional stability*: an outcome of a coalition formation process should be resistant to individual/group deviations, with different types of deviations giving rise to different notions of stability such as core stability, Nash stability, and individual stability (see the survey of Aziz and Savani [10] for an overview).

So, how do people actually communicate and form groups? Often, communication between players occurs through networks, including job markets, international alliances among countries, researchers co-authoring papers, to name a few. In the classic cooperative game setting, such restrictions of communication structure have been successfully integrated. The first pioneering work was undertaken by Myerson in 1977 [78]. Myerson [78] proposed

a transferable utility cooperative game with a restricted communication structure, given by undirected graphs. Demange [46] shows that if the underlying interaction network is a tree, then the core of a non-transferable utility game is not empty; further studies [25, 74] establish relations between approximate stability and the underlying graph structure, while Chalkiadakis et al. [36] study the computational complexity of finding core outcomes in graph restricted environments. Nevertheless, few studies have so far been made on hedonic coalition formation games where communication structure between players is restricted.

Chapter 2 develops a hedonic coalition formation game where the set of feasible coalitions is described by an underlying social network. That is, a subset of players are able to form a coalition whenever they are connected in the network. We investigate the existence and complexity issues in such games, for several notions of stability. In particular, we show that acyclic graph structure always ensures the existence of core and individually stable outcomes. Further, while computing a core stable outcome turns out to be NP-hard even for trees, we provide an efficient algorithm that finds an individually stable outcome, for an arbitrary hedonic game on an acyclic graph. We also introduce a new stability concept—*in-neighbor stability*—which is tailored for our setting. We show that the problem of finding an in-neighbor stable outcome admits a polynomial-time algorithm if the underlying graph is a path, but is NP-hard for arbitrary trees even for additively separable hedonic games; for symmetric additively separable games we obtain a PLS-hardness result.

In Chapter 3, we will consider scenarios where players form coalitions to participate in activities. A useful model for analyzing how tasks can be allocated to groups of agents is the *group activity selection problem* (GASP), proposed by Darmann et al. [42]. In GASPs, participants express preferences over pairs of the form (activity, group size). The activities are then assigned to participants so as to achieve the best performance for the whole system as well as to satisfy individual agents. The key idea behind this formulation is that ideal group size depends on the task at hand: in a company, an ideal size of the sales team may differ from that of a web developers' team. In many real-life scenarios, however, smooth communication among members of a group is crucial in order for different individuals to work together, and hence one needs to take into account communication structures among agents. For instance, a group of employers are unable to realize their full potential if no agent knows each other. We will thus study a variant of GASP, in which a group of players can only engage in the same activity if the members of the group form a connected subset of the underlying communication structure.

We show that if multiple groups can simultaneously engage in the same activity, finding a stable outcome is easy as long as the network is acyclic. In contrast, if each activity can be assigned to a single group only, stable outcomes need not exist and finding such

outcomes becomes computationally intractable, even if the underlying network is very simple: the problem of determining whether a given instance admits a core stable, Nash stable, or individually stable outcome turns out to be NP-hard if the social network is a path or a star, or if the size of each connected component is bounded by a constant. We then study the parameterized complexity of finding outcomes that are core stable, Nash stable, or individually stable. For the parameter 'number of activities', we propose an FPT algorithm for Nash stability for the case where the social network is acyclic and obtain a W[1]-hardness result for cliques (i.e., for standard GASP); similar results hold for individual stability. In contrast, finding a core stable outcome is hard even if the number of activities is bounded by a small constant, both for standard GASP and when the social network is a star. For the parameter 'number of players', all problems we consider are in XP for arbitrary social networks; on the other hand, we prove W[1]-hardness results with respect to the parameter 'number of players' for the case where the social network is a clique.

Unfortunately, a real world application often does not allow full access to the information we need. The goal of Chapter 4 is to explore the ability to learn a stable outcome against likely deviations in hedonic graph games. Assuming that players' coalitional valuations are drawn from some unknown distribution, we will try to construct *probably approximately* core stable outcomes. We show that when the interaction network is acyclic, one can efficiently compute approximately stable coalition structures; that is, player partitions that are likely to be resistant against group deviation. This result is particularly interesting as computing stable coalition structure on tree-restricted hedonic games is computationally intractable.

1.2.2 Fair Division in Structured Environments

In the second part of the thesis, our focus turns to combining networks and fair division. Fair allocation of indivisible items has been the subject of intense research in the (computational) social choice literature; we refer the reader to a survey by Bouveret et al. [27]. A central question is how to find a "fair" allocation — i.e. allocation of the goods to the players — that satisfies certain fairness desiderata. The standard approach assumes that any combination of goods is possible; however, there are settings such that connectivity constraints of an underlying network should be taken into account, e.g., fair allocation of land plots, where the graph describes the accessibility relation among the plots.

In Chapter 5, we consider fair allocation of indivisible items under an additional constraint: there is an undirected graph describing the relationship between the items, and each agent's share must form a connected subgraph of this graph. We focus on agents that have additive utilities for the items, and consider several common fair division solution concepts,

such as proportionality, envy-freeness and maximin share guarantee. While finding good allocations according to these solution concepts is computationally hard in general, we design efficient algorithms for special cases where the underlying graph has simple structure, and/or the number of agents—or, less restrictively, the number of agent types—is small. In particular, despite non-existence results in the general case, we prove that for acyclic graphs a maximin share allocation always exists and can be found efficiently.

1.3 Basic Concepts and Notation

In this section, we discuss the basic concepts that will surface throughout most of the thesis. For $s \in \mathbb{N}$, let $[s] = \{1, 2, \ldots, s\}$. For $s, t \in \mathbb{Z}$ where $s \leqslant t$, let $[s, t] = \{s, s+1, s+2, \ldots, t\}$.

1.3.1 NTU Games and Hedonic Games

The very first model of coalition formation was formulated as non-transferable utility cooperative games (NTU games) by von Neumann and Morgenstern [77]. An NTU-game describes situations in which each coalition has a feasible set of utility levels that the members of a coalition can enjoy, and players form a coalition if it is more profitable rather than being on their own. However, the utilities of players often depend only upon the composition of their coalition. In situations like this, it suffices to consider players' preferences over subgroups of players. *Hedonic games* are a subclass of NTU games, defined as a pair $(N, (\succeq_i)_{i \in N})$ where $N = [n]$ is a finite set of players and each $\succeq_i$ is a complete and transitive preference relation over the nonempty subsets of N including player i. This framework was first introduced by Drèze and Greenberg [51] and later reformulated by Banerjee et al. [16] and Bogomolnaia and Jackson [23]. The subsets of N are referred to as *coalitions*. We let $\mathcal{N}(i)$ denote the collection of all coalitions containing i. We call a coalition $S \subseteq N$ *individually rational* if $S \succeq_i \{i\}$ for all $i \in S$. Let $\succ_i$ denote the strict preference derived from $\succeq_i$, i.e., $S \succ_i S'$ if $S \succeq_i S'$, but $S' \not\succeq_i S$. Similarly, let $\sim_i$ denote the indifference relation induced by $\succeq_i$, i.e., $S \sim_i S'$ if $S \succeq_i S'$ and $S' \succeq_i S$. We will write $i : S \succ S'$ to indicate that player i strictly prefers coalition S to coalition S'; similarly, we will write $i : S \sim S'$ to indicate that player i is indifferent between coalitions S and S'.

An important subclass of hedonic games is *additively separable games*. These games model situations where each player has a specific value for every other player, and ranks coalitions according to the total value of their members [23]. Formally, a preference profile $(\succeq_i)_{i \in N}$ is said to be *additively separable* if there exists a utility matrix $U : N \times N \to \mathbb{R}$ such that for each $i \in N$ and each $S, T \in \mathcal{N}(i)$ we have $S \succeq_i T$ if and only if $\sum_{j \in S} U(i, j) \geq$

$\sum_{j \in T} U(i,j)$ [23]. Without loss of generality, we will assume that $U(i,i) = 0$ for each $i \in N$. An additively separable preference is said to be *symmetric* if the utility matrix $U : N \times N \to \mathbb{R}$ is symmetric, i.e., $U(i,j) = U(j,i)$ for all $i, j \in N$. Dimitrov et al. [49] studied a subclass of additively separable preferences, which they called *enemy-oriented preferences*. Under these preferences each player considers every other player to be either a friend or an enemy, and has strong aversion towards her enemies: $U(i,j) \in \{1, -|N|\}$ for each $i, j \in N$ with $i \neq j$.

An *outcome* of a hedonic game is a partition of players into disjoint coalitions. Given a partition π of N and a player $i \in N$, let $\pi(i)$ denote the unique coalition in π that contains i. A minimum requirement that solutions should satisfy is *individual rationality*. A partition π of N is said to be *individually rational* if all players weakly prefer their own coalitions to staying alone, i.e., $\pi(i) \succeq_i \{i\}$ for all $i \in N$.

The most commonly used stability concepts are core stable [51, 16, 23], Nash stable [23], and individually stable partitions [23]. Specifically, a coalition $S \subseteq N$ *strongly blocks* a partition π of N if $S \succ_i \pi(i)$ for all $i \in S$; it *weakly blocks* π if $S \succeq_i \pi(i)$ for all $i \in S$ and $S \succ_j \pi(j)$ for some $j \in S$. A partition π of N is said to be *core stable* (CR) if no coalition $S \subseteq N$ strongly blocks π; it is said to be *strictly core stable* (SCR) if no coalition $S \subseteq N$ weakly blocks π.

Now consider a player $i \in N$ and a pair of coalitions $S \notin \mathcal{N}(i)$, $S' \in \mathcal{N}(i)$. A player i *wants to deviate* from S' to S if $S \cup \{i\} \succ_i S'$. A player $j \in S$ *accepts* a deviation of i to S if $S \cup \{i\} \succeq_j S$. A deviation of i from S' to S is

- an *NS-deviation* if i wants to deviate from S' to S.
- an *IS-deviation* if it is an NS-deviation and all players in S accept it.

A partition π is called *Nash stable* (NS) (respectively, *individually stable* (IS)) if no player $i \in N$ has an NS-deviation (respectively, an IS-deviation) from $\pi(i)$ to another coalition $S \in \pi$ or to $\emptyset$.

We have the following containment relations among these classes of outcomes: SCR $\subseteq$ CR, SCR $\subseteq$ IS, NS $\subseteq$ IS. However, a core stable outcome need not be individually stable, and an individually stable outcome may fail to be in the core.

Without any restrictions on games, unfortunately stable outcomes may not exist as we shall see in the next example.

Example 1.1. Consider a hedonic game $(N, (\succeq_i)_{i\in N})$ where $N = \{1, 2, 3\}$, and the preferences are given as follows:

$$1 \ : \ \{1,2\} \succ \{1,3\} \succ \{1\} \succ \{1,2,3\},$$
$$2 \ : \ \{2,3\} \succ \{1,2\} \succ \{2\} \succ \{1,2,3\},$$
$$3 \ : \ \{1,3\} \succ \{2,3\} \succ \{3\} \succ \{1,2,3\}.$$

All individually rational partitions are strongly blocked by a two-player coalition and therefore not core stable. In fact, $\{\{1\}, \{2\}, \{3\}\}$ is blocked by any coalition of size two. The partition $\{\{1, 2\}, \{3\}\}$ is blocked by $\{2, 3\}$. The partition $\{\{2, 3\}, \{1\}\}$ is blocked by $\{1, 3\}$. Finally, $\{\{1, 3\}, \{2\}\}$ is blocked by $\{1, 2\}$. Moreover, $\{\{1, 2, 3\}\}$ is not individually rational. Hence, the game has empty core. Notice also that this game admits no individually stable outcome.

The stable outcomes may be empty for many classes of hedonic games; what's worse, it is in general computationally intractable to compute a stable coalition structure. Identifying a class of hedonic games whose stable outcomes are non-empty or can be efficiently computed has been thus a central topic in the literature. Sung and Dimitrov [92] were the first to consider complexity issues in additively separable hedonic games (ASHGs); they prove that it is NP-hard to determine if a game admits a core stable, strict core stable, individually stable, or Nash sable outcome (see also [81]). Aziz et al. [6] extend the first two or these results to symmetric additively separable hedonic games (SASHGs). While SASHGs always admit a Nash stable or individually stable partition [23, 32], finding one may still be difficult: Gairing and Savani [55] prove that finding such partitions is PLS-hard (PLS-hardness is a complexity class for total search problems, see [84]).

1.3.2 Graphs and Digraphs

An *undirected graph*, or simply a *graph*, is a pair (N, L), where N is a finite set of *nodes* and $L \subseteq \{\, \{i, j\} \mid i, j \in N, i \neq j \,\}$ is a collection of *edges* between nodes. Given a set of nodes S, the *subgraph of* (N, L) *induced by* S is the graph (S, L_S), where $L_S = \{\{i, j\} \in L \mid i, j \in S\}$. For a graph (N, L), a sequence of distinct nodes $(i_1, i_2, \ldots, i_k)$, $k \geq 2$, is called a *path* in L if $\{i_h, i_{h+1}\} \in L$ for $h = 1, 2, \ldots, k-1$. A path $(i_1, i_2, \ldots, i_k)$, $k \geq 3$, is said to be a *cycle* in L if $\{i_k, i_1\} \in L$. A graph (N, L) is said to be a *forest* if it contains no cycles. A subset $S \subseteq N$ is said to be *connected in* (N, L) if for every pair of distinct nodes $i, j \in S$ there is a path between i and j in L_S. The collection of all connected subsets of N in (N, L) is denoted by $\mathcal{F}_L$; also, we write $\mathcal{F}_L(i) = \mathcal{F}_L \cap \mathcal{N}(i)$. By convention, we assume that $\emptyset \notin \mathcal{F}_L$. A forest (N, L) is said to be a *tree* if N is connected in (N, L). A tree (N, L)

is called a *star* if there exists a central node $c \in N$ such that $L = \{\, \{c, j\} \mid j \in N \setminus \{c\} \,\}$. A subset $S \subseteq N$ of a graph (N, L) is said to be a *clique* if for every pair of distinct nodes $i, j \in S$ we have $\{i, j\} \in L$.

A *directed graph*, or a *digraph*, is a pair (N, T) where N is a finite set of nodes and $T \subseteq N \times N$ is a collection of *arcs* between nodes. A sequence of distinct nodes $(i_1, i_2, \ldots, i_k)$, $k \geq 2$, is called a *directed path* in T if $(i_h, i_{h+1}) \in T$ for $h = 1, 2, \ldots, k-1$. Given a digraph (N, T), let $L(T) = \{\, \{i, j\} \mid (i, j) \in T \,\}$: the graph $(N, L(T))$ is the *undirected version* of (N, T). A digraph (N, T) is said to be a *rooted tree* if $(N, L(T))$ is a tree and each node has at most one arc entering it. A rooted tree has exactly one node that no arc enters, called the *root*, and there exists a unique directed path from the root to every node of N.

Let (N, T) be a rooted tree. We say that a node $j \in N$ is a *parent* of i if $(j, i) \in T$. We denote by $\text{pr}(i)$ the unique parent of i. A node $j \in N$ is called a *descendant* of i if $j = i$ or there exists a directed path from i to j. We write

$$\text{desc}(i) = \{\, j \in N \mid j \text{ is a descendant of } i \,\}.$$

A node $i \in N$ is called a *child* of $S \subseteq N$ if $i \notin S$ and $\text{pr}(i) \in S$. We write

$$\text{ch}(S) = \{\, i \in N \mid i \notin S \text{ and } \text{pr}(i) \in S \,\}.$$

For $S = \{i\}$, we abuse the notation and write $\text{ch}(i)$ as the set of children of $\{i\}$. The *height* of a node $i \in N$ is defined inductively as follows:

$$\text{height}(i) := \begin{cases} 0 \;\text{ if } \text{desc}(i) = \{i\}, \\ 1 + \max\{\, \text{height}(j) \mid j \in \text{desc}(i) \setminus \{i\} \,\} \\ \quad \text{otherwise.} \end{cases}$$

The tree-width is a complexity measure that evaluates how tree-like the graph is. Formally, for a graph (N, L), its tree decomposition is a pair $((X_v)_{v \in V}, T)$ where (V, T) is a rooted tree and $(X_v)_{v \in V}$ a family of subsets of N, called *bags*, such that:

(i) for every $i \in N$, the set $X^{-1}(i) := \{\, v \in V \mid i \in X_v \,\}$ is nonempty and connected in $(V, L(T))$; and

(ii) for every edge $\{i, j\} \in L$ there is a vertex $v \in V$ such that $i, j \in X_v$.

The *tree width* of a tree decomposition $((X_v)_{v \in V}, T)$ of (N, L) is $\max_{v \in V}(|X_v| - 1)$.

1.3.3 Parameterized Complexity

The parmeterized complexity theory provides a refined analysis of hard algorithmic problems. The basic idea is to identify a parameter of the instance such that the inevitable combinatorial explosion can be confined as a function of this parameter. In this respect, the most favourable algorithm is an FPT algorithm; a less favourable is an XP algorithm. Specifically, a parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the *parameter*.

Definition 1.1. A parameterized problem is said to be *fixed parameter tractable* (FPT) with respect to a parameter k if each instance I of this problem can be solved in time $f(k)\mathrm{poly}(|I|)$, and to be *slice-wise polynomial* (XP) with respect to a parameter k if each instance I of this problem can be solved in time $|I|^{f(k)}$.

To obtain lower bounds, we will need the notion of a parameterized reduction and the complexity class W[1] [40]. Roughly speaking, a parameterized reduction is a variant of the standard polynomial-time reduction which retains bounds on the parameter, and W[1]-hardness rules out the existence of FPT algorithms under the Exponential Time Hypothesis. A formal definition is given as follows.

Definition 1.2. Let A and B be two parameterized problems. A parameterized reduction from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that

- (x, k) is a yes-instance of A if and only if (x', k') is a yes-instance of B;
- $k' \leqslant g(k)$ for some computable function g; and
- the running time is $f(k)|x|^{O(1)}$ for some computable function f.

Problem A is W[1]-hard if for any problem B in W[1], there is a parameterized reduction from B to A. It is widely believed that FPT $\neq$ W[1]. The problem of deciding whether a graph admits a clique of size k (CLIQUE) is a well known W[1]-complete problem when parameterized by the solution size k.

Part II

Stability in Structured Environments

Chapter 2

Hedonic Games with Graph Restricted Communication

The standard model of hedonic games does not impose any restrictions on which coalitions may form. However, in reality we often encounter constraints on coalition formation. Consider, for instance, an international network of natural gas pipelines. It seems unlikely that two cities disconnected in the network would be able to coordinate a trading agreement without any help from intermediaries. Such restrictions on communication structure can be naturally described by undirected graphs, by identifying agents with nodes, communication links with edges, and feasible coalitions with connected subgraphs. In the context of cooperative transferable utility games this model was proposed in the seminal paper of Myerson [78], and has received a considerable amount of attention since then.

In contrast, very little is known about hedonic games with restricted communication structure, though some existing results for general non-transferable utility games have implications for this setting. In particular, the result of Demange [46] concerning stability in cooperative games on trees extends to non-transferable utility games, and implies that every hedonic game whose communication structure is acyclic admits a core stable partition (we discuss this result in more detail in Section 2.2.2). However, no attempt has been made to obtain similar results for other hedonic games solution concepts, or to explore algorithmic implications of constraints on the communication structure (such as acyclicity or having a small number of connected subgraphs) for computing the core and other solutions. The goal of this chapter is to make the first step towards filling this gap.

Inspired by Demange's work, we focus on hedonic games on acyclic graphs. We consider several well-studied notions of stability for hedonic games, such as individual stability, Nash stability, core stability and strict core stability, and ask two questions: (1) does acyclicity of the communication structure guarantee the existence of a stable outcome? (2) does it lead to an efficient algorithm for computing a stable outcome, and if not, are there additional

constraints on the communication structure that can be used to obtain such an algorithm? We remark that, in general, to represent the preferences of a player in an n-player hedonic game, we need to specify $n2^{n-1}$ values, which may be problematic if we are interested in algorithms whose running time is polynomial in n. We consider two approaches to circumvent this difficulty: (a) working in the oracle model, where an algorithm may submit a query of the form (i, S, S') where S and S' are two coalitions that both contain i, and learn in unit time whether i prefers S to S', S' to S or is indifferent between them; (b) considering specific succinct representations of hedonic games, such as additively separable hedonic games [23], which can be described using $n(n-1)$ numbers.

We observe that Demange's algorithm for the core runs in time that is polynomial in the number of connected subtrees of the underlying graph (in the oracle model), and use similar ideas to obtain an algorithm for finding an outcome that is both core stable and individually stable as well as an algorithm for finding a Nash stable outcome (if it exists). The running time of these algorithms can be bounded in the same way; in particular, they run in polynomial time when the graph is a path. However, we show that when the graph is a star, finding a core stable, strictly core stable or Nash stable outcome is NP-hard, even if we restrict ourselves to very simple subclasses of additively separable hedonic games. For symmetric additively separable hedonic games, we show that the PLS-hardness result for Nash stability [55] holds even if the underlying graph is a star.

In contrast, acyclicity turns out to be sufficient for individual stability: we show that every hedonic game on an acyclic graph admits an individually stable partition, and, moreover, such a partition can be computed in time polynomial in the number of players (in the oracle model). We believe that this result is remarkable, since in the absence of communication constraints finding an individually stable outcome is hard even for (symmetric) additively separable hedonic games [92, 55], and finding a Nash stable outcome in such games remains hard even for games on stars.

2.1 Model

We first define a hedonic game restricted by an undirected graph.

Definition 2.1. A *hedonic game with graph structure*, or a *hedonic graph game*, is a triple $(N, (\succeq_i)_{i\in N}, L)$ where $(N, (\succeq_i)_{i\in N})$ is a hedonic game, and $L \subseteq \{\, \{i, j\} \mid i, j \in N, i \neq j \,\}$ is the set of communication links between players. A coalition $S \subseteq N$ is said to be *feasible* if it is connected in (N, L).

If (N, L) is a clique, a hedonic graph game $(N, (\succeq_i)_{i\in N}, L)$ is equivalent to the ordinary hedonic game $(N, (\succeq_i)_{i\in N})$.

A partition π of N is said to be *feasible* if $\pi \subseteq \mathcal{F}_L$. An *outcome* of a hedonic graph game is a feasible partition. The standard definitions of stability concepts can be adapted to graph games in a straightforward manner. Specifically, we say that a coalitional deviation is *feasible* if the deviating coalition itself is feasible; an individual deviation where player i joins a coalition S is *feasible* if $S \cup \{i\}$ is feasible. Now, we modify the standard definitions given in Section 1.3.1 by only requiring stability against feasible deviations. We use the notation (N, U, L) to denote an additively separable graph game with utility matrix $U : N \times N \to \mathbb{R}$.

In many real-life situations, when people move from one group to another, they need approvals from their contacts in the new group. Suppose, for instance, that Alice is an early-career researcher applying for academic positions in universities: her application is unlikely to be accepted if it is rejected by her prospective mentors (even if Alice expects to collaborate with several other faculty members as well). Motivated by these considerations, we will now describe a new notion of stability, which is specific to hedonic graph games.

Definition 2.2. Given a hedonic graph game $(N, (\succeq_i)_{i\in N}, L)$, we say that j is a *neighbor* of i if $\{i, j\} \in L$. A feasible deviation of a player i to $S \notin \mathcal{N}(i)$ is called

- *in-neighbor feasible* if it is NS feasible and accepted by all of i's neighbors in S.
- *IR-in-neighbor feasible* if it is in-neighbor feasible and for all $j \in S$ it holds that $S \cup \{i\} \succeq_j \{j\}$.

A feasible partition π is called *in-neighbor stable (INS)* (respectively, *IR in-neighbor stable (IR-INS)*) if no player i has an in-neighbor feasible deviation (respectively, an IR-in-neighbor feasible deviation) from $\pi(i)$ to a coalition $S \in \pi \cup \{\emptyset\}$.

Note that every INS partition is IR-INS, and each IR-INS partition is individually stable. However, the converse may not be true as we can see in the following examples.

Example 2.1 (Coalition formation in Parliament)**.** *Consider the coalition formation problem in a parliament consisting of three parties: left-wing (ℓ), centrist (c), and right-wing (r). The left-wing party ℓ and the right-wing party r cannot form a coalition without the centrist c. We describe this scenario as an additively separable graph game (N, U, L) on a path where $N = \{\ell, \mathsf{c}, \mathsf{r}\}$, $L = \{\{\ell, \mathsf{c}\}, \{\mathsf{c}, \mathsf{r}\}\}$, and the utility matrix U is given by*

$$U(\ell, \mathsf{c}) = 1, U(\ell, \mathsf{r}) = -2,$$
$$U(\mathsf{c}, \ell) = 2, U(\mathsf{c}, \mathsf{r}) = 0,$$
$$U(\mathsf{r}, \ell) = 0, U(\mathsf{r}, \mathsf{c}) = 2.$$

The resulting preference profile is as follows:

$$\begin{aligned}
\ell &: \{\ell, \mathsf{c}\} \succ \{\ell\}, \\
\mathsf{c} &: \{\ell, \mathsf{c}, \mathsf{r}\} \sim \{\ell, \mathsf{c}\} \succ \{\mathsf{c}, \mathsf{r}\} \sim \{\mathsf{c}\}, \\
\mathsf{r} &: \{\ell, \mathsf{c}, \mathsf{r}\} \sim \{\mathsf{c}, \mathsf{r}\} \succ \{\mathsf{r}\},
\end{aligned}$$

where we omit coalitions that are not individually rational or not feasible. The individually rational feasible partitions of this game are $\pi_1 = \{\{\ell, \mathsf{c}\}, \{\mathsf{r}\}\}$, $\pi_2 = \{\{\ell\}, \{\mathsf{c}, \mathsf{r}\}\}$, *and* $\pi_3 = \{\{\ell\}, \{\mathsf{c}\}, \{\mathsf{r}\}\}$. *Neither partition* π_2 *nor* π_3 *is IR-in-neighbor stable since player* c *strictly prefers to deviate to coalition* $\{\ell\}$ *and player* r *accepts it. Partition* π_1 *is IR-in-neighbor stable, but admits an in-neighbor feasible deviation of player* r *to coalition* $\{\ell, \mathsf{c}\}$*, which would then break individual rationality.*

All IR partitions in Example 2.1 are not in-neighbor stable, so the existence of in-neighbor stable outcomes is not guaranteed, even in additively separable games on paths. Similarly, an IR-INS partition maynot exist even if the underlying graph is a path; we present an example in Appendix A.1.

2.2 Acyclic Games

In general, hedonic games may fail to have partitions that are core stable or individually stable as shown in Example 1.1 of the previous chapter; consequently, this is also the case for hedonic graph games on general graphs. Now, recall that in Example 1.1, the obstruction to the existence of core stable outcomes is a vicious cycle of coalitional deviations; however, if we do not allow the coalition $\{1, 3\}$ to form, the cycle of coalitional deviations disappears, and consequently a game admits a core and individually stable partition $\pi = \{\{1\}, \{2, 3\}\}$. In this section, we shall see that indeed if the underlying communication structure does not contain such cycles, some stable partitions are guaranteed to exist irrespective of preference profiles.

2.2.1 Individual Stability

In this section, we will first show that an individually stable feasible partition exists and can be efficiently computed in a hedonic graph game whose underlying graph is a forest; we will then provide a constructive proof that in the presence of cycles the existence of an individually stable feasible partition is not guaranteed.

Theorem 2.1. *Suppose that we are given oracle access to the preference relations $\succeq_i$ of all players in a hedonic graph game $(N, (\succeq_i)_{i\in N}, L)$, where (N, L) is a forest. Then we can find an individually stable feasible outcome of the game in time polynomial in $|N|$.*

Proof. We first give an informal description of our algorithm, followed by pseudocode (Algorithm 1). If the input graph (N, L) is a forest, we can process each of its connected components separately, so we can assume that (N, L) is a tree. We choose an arbitrary node r to be the root; this transforms (N, L) into a rooted tree (N, T) with root r and determines a hierarchy of players. For each player $i \in N$, from the bottom player to the top of the hierarchy, we compute a tentative partitioning $\pi^{(i)}$ of the subtree rooted at i; we denote by $B(i)$ the coalition to which i belongs at the partition $\pi^{(i)}$. To this end, among all coalitions $B(j)$ that i's children j belong to, we identify those whose members would be willing to let i join them. Then we let i choose between his most preferred option among all such coalitions and the singleton $\{i\}$. We then check if any of the descendants of i who are adjacent to i's coalition want to join it; we let them do so if they are approved by the current coalition members. For $i \in N$ and a family of subsets $\mathcal{F} \subseteq \mathcal{N}(i)$, we set

$$\max_i \mathcal{F} = \{\, S \in \mathcal{F} \mid S \succeq_i S' \text{ for all } S' \in \mathcal{F} \,\}.$$

Given a pair of nonempty subsets $S, S' \subseteq N$, we write $S \overset{m}{\succeq} S'$ if $S \cap S' \neq \emptyset$ and $S \succeq_i S'$ for all $i \in S \cap S'$.

Algorithm 1: Finding IS partitions

input : tree (N, L), $r \in N$, oracles for $\succeq_i$, $i \in N$
output : $\pi^{(r)}$

make a rooted tree with root r by orienting all the edges in L;
initialize $\mathrm{B}(i) \leftarrow \emptyset$ and $\pi^{(i)} \leftarrow \emptyset$ for each $i \in N$;
foreach $t = 0, \ldots,$ height(r) **do**
  foreach $i \in N$ *with* height$(i) = t$ **do**
    $C(i) = \{\, j \in \mathrm{ch}(i) \mid \mathrm{B}(j) \cup \{i\} \overset{m}{\succeq} \mathrm{B}(j) \,\}$;
    choose $\mathrm{B}(i) \in \max_i(\{\{i\}\} \cup \{\, \mathrm{B}(j) \cup \{i\} \mid j \in C(i) \,\})$;
    while *there exists* $j \in \mathrm{ch}(\mathrm{B}(i))$ *such that* $\mathrm{B}(i) \cup \{j\} \succ_j \mathrm{B}(j)$ *and* $\mathrm{B}(i) \cup \{j\} \overset{m}{\succeq} \mathrm{B}(i)$ **do**
      $\mathrm{B}(i) \leftarrow \mathrm{B}(i) \cup \{j\}$
    $\pi^{(i)} \leftarrow \{\mathrm{B}(i)\} \cup \{\, \pi^{(k)} \mid k \in \mathrm{ch}(\mathrm{B}(i)) \,\}$

We will now argue that Algorithm 1 correctly identifies an individually stable partition. Our argument is based on two lemmas.

Lemma 2.1. *For every $i \in N$ and every $j \in \mathrm{ch}(\{i\})$, if every member of $B(j)$ accept's i's deviation to $B(j)$, then i weakly prefers to stay in $B(i)$ than joining the coalition $B(j)$, i.e., if $\mathrm{B}(j) \cup \{i\} \overset{m}{\succeq} \mathrm{B}(j)$, then $\mathrm{B}(i) \succeq_i \mathrm{B}(j) \cup \{i\}$.*

Lemma 2.1 follows immediately from the choice of $\mathrm{B}(i)$ in Line 6 and the stopping criterion of the **while** loop in lines 7–9.

Lemma 2.2. *For each $i \in N$, $j \in \mathrm{desc}(i)$ and all $S \in \pi^{(i)} \cup \{\emptyset\}$ there is no IS feasible deviation of j from $\pi^{(i)}(j)$ to S.*

Proof. We use induction on $\mathrm{height}(i)$. For $\mathrm{height}(i) = 0$ our assertion is trivial. Suppose that it holds for all $j \in N$ with $\mathrm{height}(j) \leq t-1$, and consider $i \in N$ with $\mathrm{height}(i) = t$. By construction, $\pi^{(i)}$ is individually rational, and by the induction hypothesis, no player $j \in \mathrm{desc}(i) \setminus \mathrm{B}(i)$ has an IS feasible deviation to a coalition in $\pi^{(i)} \setminus \{\mathrm{B}(i)\}$. Moreover, by the stopping criteria in Line 14, no player has an IS feasible deviation to $\mathrm{B}(i)$.

Now, assume towards a contradiction that there exists an IS feasible deviation of $j \in \mathrm{B}(i)$ to $\mathrm{B}(j^*)$ for some $j^* \in \mathrm{ch}(j)$. Then, j strictly prefers $\mathrm{B}(j^*) \cup \{j\}$ to $\mathrm{B}(j)$, and all the players in $\mathrm{B}(j^*)$ accepts j, namely, $\mathrm{B}(j^*) \cup \{j\} \succ_j \mathrm{B}(i)$ and $\mathrm{B}(j^*) \cup \{j\} \overset{m}{\succeq} \mathrm{B}(j^*)$. First, if $j = i$, Lemma 2.1 implies that $\mathrm{B}(i) \succeq_j \mathrm{B}(j^*) \cup \{i\}$, a contradiction. Thus, $j \neq i$.

Second, suppose that player j joins the coalition $\mathrm{B}(i)$ when $\mathrm{B}(i)$ is initialized in Line 6. Then, j belongs to the coalition $\mathrm{B}(i^*)$ for some i's child i^*. Then, the coalition $\mathrm{B}(j^*)$ is in $\pi^{(i^*)}$, since the player j^* does not belong to $\mathrm{B}(i^*)$, and j^* is a child of j. Further, the second stopping criterion of the **while** loop in Line 14 ensures that j's utility does not decrease during the execution of Algorithm 1. Thus, $\mathrm{B}(i) \succeq_j \mathrm{B}(i^*)$, and we have

$$\mathrm{B}(j^*) \cup \{j\} \succ_j \mathrm{B}(i) \succeq_j \mathrm{B}(i^*),$$

and $\mathrm{B}(j^*) \cup \{j\} \overset{m}{\succeq} \mathrm{B}(j^*)$, which means that $\pi^{(i^*)}$ admits an IS feasible deviation of j from $\mathrm{B}(i^*)$ to $\mathrm{B}(j^*)$. This contradicts the induction hypothesis.

Finally, suppose that player j joins $\mathrm{B}(i)$ during the **while** loop in Lines 7–9. Then at that point player j is made better off by leaving $\mathrm{B}(j)$ and joining $\mathrm{B}(i)$. From then on, she vetoes all candidates whose presence would make her worse off. Hence, $\mathrm{B}(i) \succ_j \mathrm{B}(j)$. However, since all players in $\mathrm{B}(j^*)$ accept j's deviation to $\mathrm{B}(j^*)$, $\mathrm{B}(j) \succeq_j \mathrm{B}(j^*) \cup \{j\}$ by Lemma 2.1. Thus, $\mathrm{B}(i) \succ_j \mathrm{B}(j^*) \cup \{j\}$, a contradiction. □

The partition $\pi^{(r)}$ is feasible by construction, so applying Lemma 2.2 with $i = r$ implies that $\pi^{(r)}$ is an individually stable feasible partition of N.

It remains to analyze the running time of Algorithm 1. Consider the execution of the algorithm for a fixed player i. Let $c = |\mathrm{ch}(i)|$, $s = |\mathrm{desc}(i)|$. Line 5 requires at most s oracle

queries: no descendant of i is queried more than once. Line 6 requires c oracle queries. Moreover, at each iteration of the **while** loop in lines 7–9 at least one player joins $\mathrm{B}(i)$, so there are at most s iterations, in each iteration we consider at most s candidates, and for each candidate we perform at most s queries. Summing over all players, we conclude that the number or oracle queries is bounded by $O(|N|^4)$. This completes the proof of the theorem. □

Theorem 2.1 provides a constructive proof that every hedonic graph game whose underlying graph (N, L) is a forest admits an individually stable feasible partition. In contrast, if (N, L) contains a cycle, the players' preferences can always be chosen so that no individually stable feasible partition exists.

Proposition 2.1. *Suppose that the graph (N, L) contains a cycle $C = \{i_1, i_2, \ldots, i_k\}$ with $k \geq 3$, $\{i_h, i_{h+1}\} \in L$ for $h = 1, 2, \ldots, k$, where $i_{k+1} := i_1$. Then, we can choose preference relations $(\succeq_i)_{i \in N}$ so that the set of IS feasible partitions of the game $(N, (\succeq_i)_{i \in N}, L)$ is empty.*

Proof. Let d be the smallest natural number that does not divide k. For each $i_h \in C$, define

$$\mathcal{C}(i_h) = \{\, S \in \mathcal{F}_L(i_h) \mid S \subseteq C, |S| \leq d \,\}.$$

Notice that $d \geq 2$, and hence $\{i_h\}, \{i_h, i_{h+1}\} \in \mathcal{C}(i_h)$. Define a hedonic graph game $(N, (\succeq_i)_{i \in N}, L)$ so that for each $i_h \in C$ we have

1. $S \succ_{i_h} S'$ for all $S, S' \in \mathcal{C}(i_h)$ such that $i_{h+1} \in S \setminus S'$;
2. $S \sim_{i_h} S'$ for all $S, S' \in \mathcal{C}(i_h)$ such that $i_{h+1} \in S \cap S'$;
3. $S \sim_{i_h} S'$ for all $S, S' \in \mathcal{C}(i_h)$ such that $i_{h+1} \notin S \cup S'$; and
4. $S \succ_{i_h} S'$ for all $S \in \mathcal{C}(i_h), S' \notin \mathcal{C}(i_h)$.

Let π be an arbitrary individually rational feasible partition of N. By individual rationality, $\pi(i) \in \mathcal{C}(i)$ for every $i \in C$. Since d does not divide k, there exists $S \in \pi$ such that $|S| \leq d - 1$ and $S \subseteq C$. Let $S = \{i_{h+1}, i_{h+2}, \ldots, i_m\}$. Then we have $\{i_h\} \cup S \succ_{i_h} \pi(i_h)$, and $\{i_h\} \cup S \sim_{i_j} S$ for all $i_j \in S$. Thus, π is not individually stable. □

We summarize our results for individually stable feasible partitions in the following corollary.

Corollary 2.1. *For the class of hedonic graph games, the following statements are equivalent.*

(i) (N, L) *is a forest.*

(ii) *For every hedonic graph game* $(N, (\succeq_i)_{i\in N}, L)$ *there exists an individually stable feasible partition of* N.

2.2.2 Core Stability

The existence of a core stable outcome is well-known for acyclic non-transferable utility games [70, 45, 46]. These results translate into the existence guarantee for core stability in acyclic games.

Theorem 2.2 ([70, 45, 46])**.** *For the class of hedonic games with graph structure, the following statements are equivalent.*

(i) (N, L) *is a forest.*

(ii) *For every hedonic graph game* $(N, (\succeq_i)_{i\in N}, L)$ *there exists a core stable feasible partition of* N.

We will now show that these two existence results for core and individual stability can be combined, in the following sense: if (N, L) is acyclic, then every hedonic game on (N, L) admits a feasible partition that belongs to the core and is individually stable; moreover, the converse is also true.

Theorem 2.3. *For the class of hedonic games with graph structure, the following statements are equivalent.*

(i) (N, L) *is a forest.*

(ii) *For every hedonic graph game* $(N, (\succeq_i)_{i\in N}, L)$*, there exists a feasible partition of* N *that is both core and individually stable.*

Proof. The direction (ii) $\Rightarrow$ (i) immediately follows from Theorem 2.2. We will prove (i) $\Rightarrow$ (ii). Recall that a preference relation $\succeq_1$ on a set X is a *refinement* of a preference relation $\succeq_2$ on X if for every $a, b \in X$ it holds that $a \succ_2 b$ implies $a \succ_1 b$.

Suppose that (N, L) is a forest. For each $i \in N$, let $>_i$ be a refinement of $\succeq_i$ that satisfies $S >_i S'$ whenever $S \sim_i S'$ and $S' \subsetneq S$. By Theorem 2.2, the game $(N, (>_i)_{i\in N}, L)$ admits a core stable feasible partition π. By construction, π is core stable in the original game $(N, (\succeq_i)_{i\in N}, L)$ as well. We will now argue that it is also individually stable. Assume towards a contradiction that there exists an IS feasible deviation of some player $i \in N$ from $\pi(i)$ to $S \in \pi \cup \{\emptyset\}$. That is, $S \cup \{i\} \in \mathcal{F}_L$, $S \cup \{i\} \succ_i \pi(i)$, and $S \cup \{i\} \succeq_j S$ for each

$j \in S$. By construction of $(>_i)_{i \in N}$, this implies that $S \cup \{i\} >_i \pi(i)$ and $S \cup \{i\} >_j S$ for each $j \in S$. This contradicts the fact that π is a core stable partition of the game $(N, (>_i)_{i \in N}, L)$. □

Among the papers showing the existence of a core stable outcome [70, 45, 46], Demange [46] provides an algorithm to find a core stable partition. This algorithm is similar in flavor to Algorithm 1: it processes the players starting from the leaves and moving towards the root, calculates the "guarantee level" of each player, and then partitions the players into disjoint groups in such a way that the final outcome satisfies their "guarantee levels"; this is shown to ensure core stability. While Demange does not analyze the running time of her algorithm, it can be verified that it runs in time polynomial in the number of connected subsets of the underlying graph. Thus, in particular, Demange's algorithm runs in polynomial time if this graph is a path.

Theorem 2.4 (implicit in [46])**.** *Suppose that we are given oracle access to the preference relations $\succeq_i$ of all players in a hedonic graph game $\mathcal{G} = (N, (\succeq_i)_{i \in N}, L)$, where (N, L) is a forest. Then we can find a core stable feasible outcome of $\mathcal{G}$ in time polynomial in the number of connected subsets of (N, L).*

Proof. Again, if our input is a forest, we can treat each connected component separately; hence, we assume that the graph is a tree. To begin with, we first construct a rooted spanning tree with root $r \in N$ of (N, L). For each player $i \in N$, starting from the leaves of the rooted tree to the root, we choose a connected subset $\mathrm{G}(i)$ of $\mathrm{desc}(i)$ that maximizes i's utility under the constraint that every descendant j of i in the coalition can agree, i.e., for any descendant $j \in \mathrm{G}(i)$ of i, $\mathrm{G}(i)$ is at least as preferred as $\mathrm{G}(j)$. The utility level of $\mathrm{G}(i)$ determined for each player $i \in N$ can be interpreted as i's guaranteed utility in the final outcome. We then inductively construct a partition $\pi^{(i)}$ of $\mathrm{desc}(i)$ that consists of $\mathrm{G}(i)$ and each $\pi^{(j)}$ of $j \in \mathrm{ch}(\mathrm{G}(i))$. The final outcome of the algorithm is a partition $\pi^{(r)}$ of the ground coalition N. We provide a formal description of the algorithm below (Algorithm 4).

We first prove that $\pi^{(r)}$ is core stable. It is easy to see that for each $i \in N$, each player $j \in \mathrm{desc}(i)$ weakly prefers $\pi^{(i)}(j)$ to their guaranteed coalition, i.e., $\pi^{(i)}(j) \succeq_j \mathrm{G}(j)$. Assume towards a contradiction that there is a connected subset S that strongly blocks $\pi^{(r)}$. Let $i = \mathrm{argmax}_{j \in S} \mathrm{height}(j)$. Then, $S \subseteq \mathrm{desc}(i)$. Since S strongly blocks $\pi^{(r)}$, we have

$$S \succ_j \pi^{(r)}(j) \succeq_j \mathrm{G}(j) \tag{2.1}$$

for all $j \in S$; in particular we have $S \succ_i \mathrm{G}(i)$. This means that by the definition of a guaranteed coalition there is a player $j \in S$ who strictly prefers $\mathrm{G}(j)$ to S, contradicting 2.1.

Algorithm 2: Finding core stable partitions

input : tree (N, L), $r \in N$, oracles for $\succeq_i$, $i \in N$
output : $\pi^{(r)}$

make a rooted tree with root r by orienting all the edges in L;
initialize $\mathrm{G}(i) \leftarrow \emptyset$ and $\pi^{(i)} \leftarrow \emptyset$ for each $i \in N$;
foreach $t = 0, \ldots, \mathrm{height}(r)$ **do**
    foreach $i \in N$ *with* $\mathrm{height}(i) = t$ **do**
        choose $\mathrm{G}(i) \in \max_i \{ S \in \mathcal{F}_L(i) \mid S \subseteq \mathrm{desc}(i) \wedge S \succeq_j \mathrm{G}(j) \text{ for all } j \in S \}$;
        $\pi^{(i)} \leftarrow \{\mathrm{G}(i)\} \cup \{ \pi^{(j)} \mid j \in \mathrm{ch}(\mathrm{G}(i)) \}$

The number of queries is bounded by $O(n^2|\mathcal{F}_L|)$ since there are at most n iterations, for each iteration we check at most $|\mathcal{F}_L|$ coalitions, and for each coalition we compare with at most n guaranteed coalitions. This completes the proof. □

Combining Demange's algorithm with the construction in the proof of Theorem 2.3, we obtain an algorithm that has the same worst-case running time as Demange's algorithm and outputs a feasible partition that belongs to the core and is individually stable.

Corollary 2.2. *Suppose that we are given oracle access to the preference relations $\succeq_i$ of all players in a hedonic graph game $\mathcal{G} = (N, (\succeq_i)_{i \in N}, L)$, where (N, L) is a forest. Then we can find a feasible outcome of $\mathcal{G}$ that belongs to the core and is individually stable in time polynomial in the number of connected subsets of (N, L).*

Nevertheless, if the number of connected subsets of (N, L) is super-polynomial in $|N|$, so is the running time of Demange's algorithm, because for each player i this algorithm considers all feasible coalitions containing i. Now, for many n-node trees the number of connected subtrees is superpolynomial in n: for instance, this is the case for every tree with $\omega(\log n)$ leaves, simply because we can delete any subset of leaves and still obtain a connected graph. Therefore, it is natural to ask if checking all feasible coalitions is indeed necessary. Note that when the goal is to find an individually stable partition, the answer to this question is "no": Algorithm 1 only considers some of the feasible coalitions, yet is capable of finding an individually stable feasible outcome. In contrast, for the core it seems unlikely that one can obtain a substantial improvement over the running time of Demange's algorithm: our next theorem shows that finding a core stable feasible partition of an additively separable hedonic graph game is NP-hard even if the underlying graph is a star and even if the input game is a symmetric enemy-oriented game.

Theorem 2.5. *If one can find a core stable feasible partition in a symmetric enemy-oriented graph game whose underlying graph is a star in time polynomial in the number of players then* P = NP.

Proof. We provide a reduction from the NP-complete CLIQUE problem [57]. An instance of CLIQUE is a pair (G, t), where G is an undirected graph and t is a positive integer. It is a "yes"-instance if G contains a clique of size at least t and a "no"-instance otherwise. We will show how a polynomial-time algorithm for our problem can be used to decide CLIQUE in polynomial time.

Given an instance (G, t) of CLIQUE, where $G = (V, E)$, we construct a symmetric enemy-oriented graph game as follows. We let $N = V \cup \{s\}$ and $L = \{\, \{s, v\} \mid v \in V \,\}$. We will now describe the (symmetric) matrix U. Briefly, player s likes all other players and two players in V like each other if and only if they are connected by an edge of G. Formally, we set $U(s, v) = 1$ for each $v \in V$ and for each $u, v \in V$ we set $U(u, v) = 1$ if $\{u, v\} \in E$ and $U(u, v) = -|V| - 1$ otherwise.

Let π be an individually rational feasible partition of this game. Note that all players in $N \setminus \pi(s)$ form singleton coalitions in π, and $\pi(s) \setminus \{s\}$ is a clique in G. We will now argue that π is core stable if and only if $\pi(s) \setminus \{s\}$ is a maximum-size clique in G.

Indeed, if C is a maximum-size clique in G and $|\pi(s) \setminus \{s\}| < |C|$, every player in $C \cup \{s\}$ strictly prefers $C \cup \{s\}$ to its current coalition. Conversely, suppose that $\pi(s) \setminus \{s\}$ is a maximum-size clique, yet coalition X strictly blocks π. Then it has to be the case that $s \in X$, and hence $|X| > |\pi(s)|$; but this means that $X \setminus \{s\}$ is not a clique, and therefore players in $X \setminus \{s\}$ prefer π to X, a contradiction.

It follows that, by looking at a core stable feasible outcome π, we can decide whether G contains a clique of size at least t. □

Dimitrov et al. [49] show that finding a core outcome in symmetric enemy-oriented games is NP-hard; however, in their model there is no constraint on communication among the players, i.e., their result is for the case where (N, L) is a clique, whereas our result holds even if (N, L) is a star.

2.2.3 Strict Core Stability

Unlike core stable outcomes, strictly core stable outcomes need not exist even in symmetric enemy-oriented games on stars. Consider, for instance, a variant of our parliamentary coalition formation example (Example 2.1) where the centrist party (c) is equally happy to collaborate with the left-wing party (ℓ) or the right-wing party (r), but the left-wing party and the right-wing party hate each other. This setting can be captured by a symmetric

enemy-oriented graph game whose underlying graph is a path, and whose core stable feasible partitions are $\pi_1 = \{\{\ell, \mathsf{c}\}, \{\mathsf{r}\}\}$ and $\pi_2 = \{\{\ell\}, \{\mathsf{c}, \mathsf{r}\}\}$. However, neither π_1 nor π_2 is in the strict core: π_1 is weakly blocked by $\{\mathsf{c}, \mathsf{r}\}$ and π_2 is weakly blocked by $\{\ell, \mathsf{c}\}$.

Our next theorem shows that checking whether a given symmetric enemy-oriented hedonic graph game admits a feasible outcome in the strict core is NP-hard with respect to Turing reductions, even if the underlying graph is a star.

Theorem 2.6. *If there exists a polynomial-time algorithm that, given a symmetric enemy-oriented graph game whose underlying graph is a star, decides whether this game has a strictly core stable feasible partition then* P = NP.

Proof. We define UNIQUE CLIQUE as the decision problem of determining whether a graph has a unique maximum-size clique. That is, let t be the size of a maximum clique in G; G is a "yes"-instance of UNIQUE CLIQUE if it contains exactly one clique of size t and a "no"-instance otherwise. CLIQUE admits a Turing reduction to UNIQUE CLIQUE. Specifically, given an instance (G, k) of CLIQUE where $G = (V, E)$, we construct $|V|$ instances of UNIQUE CLIQUE as follows. For each $t = 1, \ldots, |V|$, let C_t be a set of size t with $V \cap C_t = \emptyset$, and let $H_t = (V \cup C_t, E_t)$, where $\{u, v\} \in E_t$ if and only if $u, v \in C_t$, or $u, v \in V$ and $\{u, v\} \in E$. Note that the maximum clique size in G is r if and only if H_r is a "no"-instance of UNIQUE CLIQUE, but H_t is a "yes"-instance of UNIQUE CLIQUE for $t = r + 1, \ldots, |V|$. Hence, a polynomial-time algorithm for UNIQUE CLIQUE can be used to decide CLIQUE in polynomial time.

We will now argue that UNIQUE CLIQUE can be reduced to our problem. Given an undirected graph $G = (V, E)$, we construct the same symmetric enemy-oriented graph game (N, U, L) as in the proof of Theorem 2.5. Then, one can readily see that the G is a "yes"-instance of UNIQUE CLIQUE if and only if (N, U, L) has a feasible outcome that is strictly core stable. □

For the broader class of symmetric additively separable hedonic graph games on stars, we obtain an NP-hardness result under the more standard notion of a many-one reduction (for games on cliques, this follows from the results of Aziz et al. [6]).

Theorem 2.7. *Given a symmetric additively separable hedonic graph game whose underlying graph is a star, it is* NP*-hard to determine whether it has a strictly core stable feasible partition.*

Proof. Again, we provide a reduction from CLIQUE. Given an undirected graph $G = (V, E)$ and a positive integer $k \geqslant 2$, we construct a symmetric additively separable graph game (N, U, L) as follows. We create a vertex player v for each $v \in V$, and three additional

players a, b, and c. We then create a star with the center b, namely, $N = \{a, b, c\} \cup V$ and $L = \{\{a, b\}, \{b, c\}\} \cup \{\, \{b, v\} \mid v \in V \,\}$. Now let $M = |N| + 1$. The utility matrix $U : N \times N \to \mathbb{R}$ is given as follows (see Figure 2.1):

$$
\begin{aligned}
&U(a, b) = U(c, b) = k - 1, U(a, c) = -M, \\
&U(a, v) = U(c, v) = -M, U(b, v) = 1 \text{ for each } v \in V, \\
&U(u, v) = -1/(k - 1) \text{ if } \{u, v\} \in E \\
&\text{and } U(u, v) = -M \text{ otherwise, for all } u, v \in V.
\end{aligned}
$$

Intuitively, players a, b, and c forms an empty strict core instance: player b equally likes a and c, but the players a and c hate each other. Each vertex player v likes the center b but dislikes the other players but v can be together with b and at most $k - 1$ other vertex players who are adjacent to v.

Suppose that G contains a clique C of size k. We will show that the partition $\pi = \{\{a\}, \{c\}, C \cup \{b\}\} \cup \{\, \{v\} \mid v \in V \setminus C \,\}$ is strictly core stable. It is immediate to see that π is individually rational. None of the pairs $\{a, b\}$ and $\{b, a\}$ weakly blocks π since $\sum_{v \in C} U(b, v) = k > k - 1$. Also, there is no weakly blocking coalition that contains both a and c, or one of the players a and b together with a vertex player, since some of such players hate each other. If a connected coalition $S \subseteq V \cup \{b\}$ weakly blocks π, then $b \in S$ by the connectivity of S, and hence $|S| > |\pi(b)|$; but this means that S includes more than k vertex players, and players in $S \setminus \{s\}$ strictly prefer $\pi(b)$ to S, a contradiction.

Conversely, suppose that there exists a strictly core stable feasible partition π of N. We will show that $\pi(b) \setminus \{b\}$ forms a clique of size at least k in G. By individual rationality of π, $\pi(b)$ cannot include a vertex player and a player from $\{b, c\}$; also, $\pi(b)$ is not contained in $\{a, b, c\}$ since otherwise π would not be strictly core stable. Hence, $\pi(b)$ consists of b and vertex players. If $|\pi(b) \setminus \{b\}| < k$, then the pair $\{a, b\}$ weakly blocks π; hence $|\pi(b) \setminus \{b\}| \geq k$. Moreover, if $\pi(b) \setminus \{b\}$ includes a pair of non-adjacent vertices, π is not individually rational, a contradiction. Hence any pair of vertex players in $\pi(b) \setminus \{b\}$ are adjacent. We conclude that $\pi(b) \setminus \{b\}$ forms a clique of size at least k. □

2.2.4 Nash, In-neighbor, and IR-in-neighbor Stability

Having observed that acyclic games admits a polynomial time algorithm to find an individually stable partition, we will consider the complexity of computing stronger solution concepts resistant to individual deviations, such as Nash stability. The following result shows that we should not hope to obtain a polynomial-time algorithm for finding such stable outcomes, even for additively separable hedonic graph games on stars.

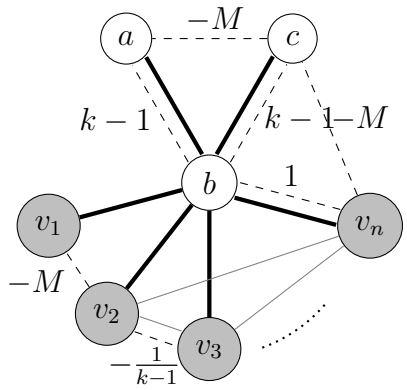


Figure 2.1: Graph used in the proof of Theorem 2.7. Thick black lines represent communication links between players, whereas gray lines stand for edges of the given instance G. Values on dashed lines are utilities of players.

Theorem 2.8. *The problems of deciding the existence of a Nash stable or an in-neighbor stable partition is* NP*-complete for an additively separable hedonic graph game whose underlying graph is a star.*

Proof. Nash and in-neighbor stability can be verified in polynomial time, so our problems are in NP. To prove NP-hardness, we reduce from CLIQUE.

Given an undirected graph $G = (V, E)$ and a positive integer k, we construct an additively separable hedonic graph game (N, U, L) on a star as follows. We create a vertex player v for each $v \in V$ and three additional players a, b, and c; we then create a star with the center b, i.e., $N = V \cup \{a, b, c\}$, and $L = \{\, \{b, v\} \mid v \in V \,\} \cup \{\{a, b\}, \{b, c\}\}$.

Intuitively, the players a, b and c will form an empty INS instance; two players a and c dislike any vertex player while the center player b has a utility 1 for such a player. Formally, let $M = |N| + 1$ and define utilities for these players as follows.

$$U(a, b) = 1, U(a, c) = -2, U(b, a) = k,$$
$$U(b, c) = 0, U(c, a) = 0, U(c, b) = 2,$$
$$U(a, v) = U(c, v) = -M \text{ for each } v \in V,$$
$$U(b, v) = 1 \text{ for each } v \in V,$$

Each vertex player dislikes players a and c, as well as their non-neigbhours. Specifically, the utilities of each vertex $v \in V$ are given as follows.

$$U(v, a) = U(v, c) = -M, U(v, b) = 0$$
$$U(v, u) = 0 \text{ if } \{u, v\} \in E \text{ and } U(v, u) = -M \text{ otherwise, for all } u, v \in V.$$

We will now show that G contains a clique of size k if and only if the game admits an in-neighbor stable feasible partition.

Suppose first that G contains a clique C of size k. Let $\pi = \{\{a\}, \{c\}, C \cup \{b\}\} \cup \{\, \{v\} \mid v \in V \setminus C \,\}$. Clearly, players a and c do not want to deviate to another coalition. Also, no player $v \in V$ can profitably deviate. Player b does not want to deviate to $\{a\}$, $\{c\}$, or any singleton in $V \setminus C$ because $\sum_{v \in C} U(b, v) \geq k \geq 1$. Thus, π is in-neighbor stable.

Conversely, suppose that π is an in-neighbor stable feasible partition of N. Then b is not together with a or c since this would cause in-neighbor feasible deviations by at least one of the three players. Thus, in π player b is grouped together with players in V only. Then, by in-neighbor stability $|\pi(b) \cap V| \geqslant k$, and by individual rationality $\pi(b) \cap V$ is a clique in G.

The same reduction shows that it is hard to find a Nash stable outcome (for additively separable hedonic games with unrestricted communication, this was shown by Sung and Dimitrov [92]). □

A similar reduction shows that it is NP-complete to decide whether an acyclic game admits an IR-in-neighbor stable partition. The proof can be found in Appendix A.1.

Theorem 2.9. *The problems of deciding the existence of an IR-in-neighbor stable partition is* NP*-complete for an additively separable hedonic graph game whose underlying graph is a tree.*

In contrast, for any hedonic game on a star, we can construct an IR-in-neighbor stable partition efficiently.

Proposition 2.2. *Every hedonic graph game $(N, (\succeq_i)_{i \in N}, L)$ where (N, L) is a star has an IR-in-neighbor stable partition, and given oracle access to the players' preference relations, such a partition can be found using $O(|N|^3)$ oracle calls.*

Proof. If the central node strictly prefers being on her own, rather than being in any coalition of size two, a partition with all singletons is IR-in-neighbor stable. Otherwise, choose a favorite two-player coalition of the center, and keep adding players to this coalition one by one if this deviation is IR-in-neighbor feasible. When no player can be added, the resulting partition is IR-in-neighbor stable, since the utility of the central node does not decrease during the execution, and there is no player who can IR-in-neighbor deviate to the coalition of the center. The bound on the running time is immediate. □

The construction in the proof of Theorem 2.8 uses an additively separable hedonic game that is not symmetric. Indeed, Bogomolnaia and Jackson [23] observe that in symmetric additively separable hedonic games, any NS deviation strictly increases the sum of all

players' utilities $\sum_{i \in N} \sum_{j \in \pi(i)} U(i,j)$. This implies that for this class of games a sequence of NS deviations converges to a Nash stable outcome. Thereby, the set of Nash stable outcomes (and hence also IS outcomes) is always non-empty. However, the number of deviations needed to reach a Nash stable outcome may be exponential in the number of players, so it remains unclear if a Nash stable outcome can be computed efficiently.

The complexity class that appears to be useful for capturing the complexity of this problem is PLS (Polynomial Local Search) [64]. A problem in PLS consists of a finite set of candidate solutions, each of which has associated neighborhood and cost. It is specified by three polynomial-time algorithms. The first algorithm computes an initial candidate solution (e.g. the all-singleton partition). The second algorithm returns the cost of each candidate solution (e.g. the social welfare of a partition). Finally, the third algorithm tests whether a given candidate solution is optimal in its neighborhood, and if not, finds a solution with better cost (e.g. an improved partition after a profitable deviation). Given two PLS problems A and B, we say that A is *PLS-reducible* to B if there exist polynomial time computable functions f and g such that f maps instances of A to B and g maps the local optima of B to local optima of A.

Gairing and Savani show that search problems related to NS and IS for symmetric additively separable games are PLS-complete [55, 56]. However, if one were to interpret the hedonic game in their reduction as a hedonic graph game, the underlying graph would necessarily contain cycles. In what follows, we will show that computing in-neighbor stable outcomes for symmetric additively separable games is PLS-complete even when the graph (N, L) is a star.

Theorem 2.10. *The problems of finding a Nash stable or an in-neighbor stable partition are* PLS*-complete for a symmetric additively separable hedonic graph game whose underlying graph is a star.*

Proof. With symmetric additively separable preferences, we are able to test local optimality of a partition and, if it is not optimal, find an improving move in polynomial time. Specifically, to prove that our problem is in PLS, we need to describe the following three algorithms.

- The first algorithm takes as an input a symmetric additively separable hedonic graph game, and returns a partition that consists of a singleton of each player.
- The second algorithm takes as an input a symmetric additively separable hedonic graph game and a feasible partition π, and returns the *social welfare*, $\sum_{i \in N} \sum_{j \in \pi(i)} U(i,j)$.

- The third algorithm checks whether or not the given feasible partition is in-neighbor stable (respectively, Nash stable); if so, it reports "local optimal;" if not, it finds a player $i \in$ who has an INS (respectively, NS) feasible deviation to some coalition $S \in \pi \cup \{\emptyset\}$ and returns the partition $\pi' = (\pi \setminus \{S, \pi(i)\}) \cup \{S \cup \{i\}, \pi(i) \setminus \{i\}\}$, which has a greater social welfare.

Clearly, the algorithms described above run in polynomial time in the description size of the given game, and hence our problems are in PLS.

To prove PLS-hardness for in-neighbor stability, we provide a reduction from LOCAL MAX-CUT, which is known to be PLS-complete [84]. Recall that an instance of LOCAL MAX-CUT is given by a weighted graph $G = (V, E, w)$, where $w : V \times V \to \mathbb{N}$ is the weight function with the convention that $w(u, v) = 0$ for each $\{u, v\} \notin E$. A *cut* is a partition of V into two parts S and $V \setminus S$; its *weight* is given by $\sum_{u \in S, v \in V \setminus S} w(u, v)$. The neighborhood of a cut $(S, V \setminus S)$ is defined as the set of all cuts that can be obtained by moving one node from S to $V \setminus S$ or vice versa; the goal is to find a cut that has the maximum weight in its neighborhood.

Given an instance (V, E, w) of LOCAL MAX-CUT, we set $N = V \cup \{s\}$, $L = \{\, \{s, v\} \mid v \in V \,\}$ and construct a symmetric additively separable graph game (N, U, L) with the utility matrix $U : N \times N \to \mathbb{R}$ defined as follows. We set $U(u, s) = \sum_{v \in V} w(u, v)$ for each $u \in V$. For every pair of distinct nodes $u, v \in V$, we define $U(u, v) = -2w(u, v)$ if $\{u, v\} \in E$ and $U(u, v) = 0$ otherwise. Note that every $u \in V$ will be in a coalition with s or on her own in any feasible partition. Moreover, player s will accept all in-neighbor feasible deviations since her utility for every other player is non-negative.

Let π be an in-neighbor stable feasible partition of N. Let $V_1 = \{\, u \in V \mid u \in \pi(s) \,\}$ and $V_2 = V \setminus V_1$. Each player $u \in V_1$ has non-negative utility $U(u, s) + \sum_{v \in V_1} U(u, v)$ by individual rationality. Hence, $\sum_{v \in V_1} w(u, v) \leq \sum_{v \in V_2} w(u, v)$ for every $u \in V_1$. On the other hand, no player $u \in V_2$ wants to deviate to $\pi(s)$: by a similar calculation, $\sum_{v \in V_1} w(u, v) \geq \sum_{v \in V_2} w(u, v)$ for every $u \in V_2$. Thus, (V_1, V_2) is a local max-cut of G.

The above construction can also be used to show PLS-completeness of finding a Nash stable partition in this class of games. □

However, we cannot extend Theorem 2.10 to enemy-oriented games where players cannot express their preferences in varying degrees.

Proposition 2.3. *A Nash stable feasible outcome of a symmetric enemy-oriented game on a star can be computed in polynomial time.*

Proof. We initialize coalition S to the center of the star; as long as there is a player that likes (and is liked by) all current members of S, we add him to S. Eventually, no player can be added to S. At this point, $\{S\} \cup \{\{i\} \mid i \in N \setminus S\}$ is a Nash stable feasible partition: no player outside of S wants to deviate to S, and no player in S wants to leave. □

2.3 Beyond Trees

So far we have studied the complexity questions when the social network is a tree. Thus, it is interesting to see if our algorithms can be extended to graphs that are "almost" acyclic. To this end, we focus on IRLC (individually rational lists of coalitions) games where players have a possibly exponential preference list of individually rational coalitions. Remarkably, the computation of core stable outcomes become easy under this representation if the graph is a tree. In what follows, we will show that for some stable outcomes, the problems remain still tractable if the graph is almost acyclic. We note that the existence of such an algorithm for arbitrary graphs is unlikely; Ballester [15] showed that computing Nash, core, and individually stable outcomes of the game is NP-hard for IRLC games and anonymous games with unrestricted communication structure.

We will show that it is easy to decide whether an IRLC game admits an outcome that is immune to individual deviations, if the graph has a bounded tree-width.

Theorem 2.11. *Suppose that we are given oracle access to the preference relations $\succeq_i$ of all players in a hedonic graph game $(N, (\succeq_i)_{i\in N}, L)$, together with a tree decomposition $((X_v)_{v\in V})$ of the underlying graph (N, L) with root r and width tw. Then we can decide whether $\mathcal{G}$ admits a Nash stable, in-neighbor stable, IR-in-neighbor stable, or individually stable feasible outcome (and find one if it exists) in time polynomial in $O(|N|^2|V|^2|\mathcal{I}|^{2tw+2})$ where $\mathcal{I}$ is the set of individually rational feasible coalitions of the game.*

Proof. For each $v \in V$, let Π_v denote the family of a set π of individually rational feasible coalitions such that π is pairwise disjoint, every coalition in π intersects with X_v, and π covers X_v, i.e., $X_v \subseteq \bigcup_{S\in\pi} S$; we denote by $\pi(i)$ the unique coalition containing each $i \in \bigcup_{S\in\pi} S$. Notice that we have $|\Pi_v| \leq |\mathcal{I}|^{tw+1}$ for each $v \in V$. For each $\pi \in \Pi_v$ and $\pi' \in \Pi_u$ with $v, u \in V$, we say that π and π' are *compatible* if for any $i, j \in X_v \cap X_u$, $\pi(i) = \pi(j)$ if and only if $\pi'(i) = \pi'(j)$. Also, we let $Y_v = \bigcup_{u\in\mathsf{desc}(v)} X_u$ for each $v \in V$.

Fix $\alpha \in \{\text{NS, INS, IR-INS}\}$. We give a dynamic programming for computing α feasible outcomes in a bottom-up manner. For each $v \in V$ and each $\pi \in \Pi_v$, we determine whether there exists an individually rational feasible partition π^* of $Y_v \cup \bigcup_{S\in\pi} S$ such that $\pi \subseteq \pi^*$ and no player can deviate to coalitions $S \in \pi^*$ under given stability requirement, namely,

there is no player $i \in Y_v \cup \bigcup_{S\in\pi} S$ and coalition $S' \in \pi^*$ such that the deviation of i from $\pi^*(i)$ to S' is α feasible. We define $\alpha[v,\pi] = 1$ if such a partition exists, and $\alpha[v,\pi] = 0$ otherwise. An α feasible solution exists if and only if $\alpha[r,\pi] = 1$ for the root r and some $\pi \in \Pi_r$. To this end, we check for each $v \in V$ whether no player wants to deviate among coalitions in π and whether for each $u \in \mathrm{ch}(v)$ there exists a family π_u of coalitions that is compatible with π and $\alpha[u,\pi_u] = 1$. Algorithm 3 describes in detail how $\alpha[v,\pi]$ is computed.

Algorithm 3: Determining the existence of α feasible partitions, where $\alpha \in$ {NS, INS, IR-INS}

input : a tree decomposition $((X_v)_{v\in V}, T)$ of (N, L) and oracles for $\succeq_i$, $i \in N$
output : $\alpha[v,\pi] \in \{0,1\}$ for $v \in V$ and $\pi \in \Pi_v$

```
1  initialize α[v,π] ← 1 for v ∈ V and π ∈ Π_v;
2  foreach t = 0, ..., height(r) do
3      foreach v ∈ V with height(v) = t do
4          foreach π ∈ Π_v do
5              if there exist a player i ∈ ⋃_{S∈π} S and a coalition S' ∈ π ∪ {{i}} where
               the deviation of i from π(i) to S' is α feasible then
6                  α[v,π] ← 0;
7              else
8                  foreach u ∈ ch(v) do
9                      if for each π' ∈ Π_u, π' is not compatible with π or α[u,π'] = 0
                       then
10                         α[v,π] ← 0;
```

Lemma 2.3. *For each $\alpha \in$ {NS, INS, IR-INS}, each $v \in V$, and each $\pi \in \Pi_v$, we have $\alpha[v,\pi] = 1$ if and only if there exists an α feasible partition π^* of $Y_v \cup \bigcup_{S\in\pi} S$ such that $\pi \subseteq \pi^*$.*

Proof. The proof is by induction on height(v). The claim is immediate when height$(v) = 0$. Suppose that it holds for all $u \in N$ with height$(v) \leq t-1$, and consider $v \in V$ with height$(v, A^r) = t$ and an arbitrary $\pi \in \Pi_v$.

Suppose first that $\alpha[v,\pi] = 1$. Line 5 ensures that each coalition in π is individually rational and no player has an α feasible deviation among coalitions in π. Since $\alpha[v,\pi] = 1$, Line 9 ensures that for each $u \in \mathrm{ch}(v)$ there exists a set $\pi_u \in \Pi_u$ of coalitions such that $\alpha[u,\pi_u] = 1$ and π_u is compatible with π. By the induction hypothesis, for each $u \in \mathrm{ch}(v)$ there exists an α feasible partition π_u^* of $Y_u \cup \bigcup_{S\in\pi_u} S$ such that $\pi_u \subseteq \pi_u^*$; combining these partitions with π, we obtain a desired partition.

Conversely, if $\alpha[v, \pi] = 0$, then at least one of the conditions of the **if** statements in Line 5 and Line 9 is satisfied. In either case, there is no α feasible partition containing π. □

Lemma 2.3 immediately implies that the input game admits an α stable feasible partition if and only if $\alpha[r, \pi] = 1$ for some $\pi \in \Pi_r$. If this is the case, such a feasible partition can be found using standard dynamic programming techniques.

It remains to analyze the running time of our algorithm. Let $n = |N|$, $s = |V|$, and $m = |\mathcal{I}|^{tw+1}$. For each $v \in V$, the algorithm considers each family $\pi \in \Pi_v$ of coalitions exactly once. To check the conditions in Line 5, it makes at most n^2 oracle calls. Further, each v has at most $|V|$ children. For each child u, the algorithm considers at most m candidate set π_u of coalitions. We conclude that our algorithm performs at most $O(sc(n^2 + s \cdot m)) = O((smn)^2)$ oracle calls. □

For core-related questions, the existence of a core stable outcome is not guaranteed even for single cycles [23]. However, none of the existing hardness results, including the NP-hardness of computing a stable matching for a stable roommate instance [83], provides the lower bound for this problem; the reductions necessarily contain an unbounded number of cycles of a communication structure. We conjecture that deciding the existence of a core outcome under the IRLC representation remains hard even if the social network has a bounded tree-width.

2.4 Summary and Related Work

We have explored the existence and computational complexity of stable partitions in hedonic games on graphs. We obtained a number of algorithmic results in the general oracle-based framework, thereby showing that acyclicity of the communication network has important implications for stability. We summarize our results in Table 2.1.

We believe that our work opens up an interesting line of research, with many problems left open to explore. In particular, we have seen that some classes of hedonic games admit a stable partition; however, it is unclear how players actually reach stability. Namely, is there any natural convergence process to arrive at stable outcomes? One could, for instance, ask whether a better response dynamics on trees can converge to individual stability.

While the graph-restricted model is sufficiently general to encompass theoretical insights about stable outcomes, it might not be satisfactory to fully understand the impact of communication structures: there are other classes of communication structures that ensure the existence of stable outcomes. Such examples include stable marriage problems whose feasible coalitions can not be represented by connected subsets of an undirected graph. It

would be interesting to investigate how the existence and computation guarantees can be extended to the general framework.

Finally it remains unknown whether deciding the existence of core stable partitions is polynomial-time solvable when the set of feasible coalitions are the connected subsets of a single cycle. We expect that the crucial obstacle is how to detect a vicious cycle that prevents the existence of core stable outcomes. Also, what remains open is the complexity of computing a strictly core stable partition for a hedonic game on a path.

		IRLC	additive	s-additive	s-enemy
Individual stability	forests	P (Th.2.1)	P (Th.2.1)	P	P
IR-in neighbor stability	forests	P (Th. 2.11)	NP-c.		
	stars	P	P (Prop. 2.2)	P	P
	paths	P	P (Th. 2.11)	P	P
In-neighbor stability	forests	P (Th. 2.11)	NP-c.	PLS-c.	
	stars	P	NP-c. (Th. 2.8)	PLS-c. (Th. 2.10)	P
	paths	P	P (Th. 2.11)	P	P
Nash stability	forests	P (Th. 2.11)	NP-c.	PLS-c.	
	stars	P	NP-c. (Th. 2.8)	PLS-c. (Th. 2.10)	P (Prop.2.3)
	paths	P	P (Th. 2.11)	P	P
Core stability	forests	P ([46])	NP-h*	NP-h*	NP-h*
	stars	P	NP-h*	NP-h*	NP-h* (Th. 2.5)
	paths	P	P ([46])	P	P
Strict core stability	forests		NP-h.	NP-h.	NP-h*
	stars		NP-h.	NP-h. (Th. 2.7)	NP-h* (Th. 2.6)
	paths				

Table 2.1: Overview of complexity results. The hardness result marked with * holds with respect to Turing reductions. When no reference is given, the result follows trivially from other results in the table.

Related work The running time of some of our algorithms is polynomial in the number of connected coalitions; see the work of Elkind [52] for a characterization of graph families for which this quantity is polynomial in the number of nodes. Some papers [20, 21, 80] use the phrase "hedonic game on a tree" to refer to a hedonic game where each player i has a value $U(i,j)$ for every other player j, and pairs $\{i,j\}$ such that $U(i,j) \neq 0$ or $U(j,i) \neq 0$ form a tree; the preference relation of player i is computed based on the values $U(i,j)$: the value of a coalition S, $i \in S$, could be $\sum_{j \in S \setminus \{i\}} U(i,j)$ (this corresponds to ASHGs) or $\frac{1}{|S|} \sum_{j \in S \setminus \{i\}} U(i,j)$ (such games are known as *fractional hedonic games*). This framework is different from ours: we allow preferences that are not derived from values assigned to individual players, and in the additively separable case we allow non-adjacent players to have a non-zero value for each other.

Chapter 3

Group Activity Selection on Social Networks

Companies assign their employees to different departments, large decision-making bodies split their members into expert committees, and university faculty form research groups: division of labor, and thus group formation, is everywhere. For a given assignment of agents to activities (such as management, product development, or marketing) to be successful, two considerations are particularly important: the agents need to be capable to work on their activity, and they should be willing to cooperate with other members of their group.

In this chapter, we will consider the *group activity selection problem* (GASP), introduced by Darmann et al. [42]. In GASP, players have preferences over pairs of the form (activity, group size). The intuition behind this formulation is that certain tasks are best performed in small or large groups, and agents may differ in their preferences over group sizes; however, they are indifferent about other group members' identities. In the analysis of GASP, desirable outcomes correspond to *stable* and/or *optimal* assignments of players to activities, i.e., assignments that are resistant to player deviations and/or that maximize the total welfare.

The basic model of GASP ignores the relationships among the agents: Do they know each other? Are their working styles and personalities compatible? Typically, we cannot afford to ask each agent about her preferences over all pairs of the form (coalition, activity), as the number of possible coalitions grows quickly with the number of agents. A more practical alternative is to adopt the ideas of Chapter 2 and assume that the relationships among the agents are encoded by a *social network*, i.e., an undirected graph where nodes correspond to players and edges represent communication links between them; one can then require that each group is connected with respect to this graph.

We extend the basic model of GASP to take into account the agents' social network (gGASP). We formulate several notions of stability for this setting, including Nash stability and core stability, and study the complexity of computing stable outcomes in our model;

these notions of stability were applied in the GASP setting by Darmann et al. [42] and Darmann [41].

In the context of hedonic games, we have seen that if the underlying network is acyclic, stable outcomes are guaranteed to exist and some of the problems known to be computationally hard for the unrestricted setting become polynomial-time solvable. We obtain a similar result for GASP, but only if several groups of agents can simultaneously engage in the same activity, i.e., if the activities are *copyable*. In contrast, we show that if each activity can be assigned to at most one coalition, a stable outcome may fail to exist, and moreover finding them is computationally hard even if the underlying network is very simple. Specifically, checking the existence of Nash stable, individually stable, or core stable outcomes turns out to be NP-hard even for very restricted classes of graphs, including paths, stars, and graphs with constant-size connected components. This result stands in sharp contrast to the known computational results in the literature; indeed, in the context of of cooperative games, such restricted networks usually enable one to design efficient algorithms for computing stable solutions (see, e.g., Chalkiadakis et al. [36] and Elkind [52]).

Given these hardness results, we switch to the fixed parameter tractability paradigm. In the context of GASP, a particularly relevant parameter is the number of activities: generally speaking, we expect the number of available activities to be small in many practical applications. For instance, companies can only assign a limited number of projects to their employers; a workshop can usually organise a couple of social events; and schools can offer few facilities to their students. We show that the problem of deciding the existence of Nash stable outcomes for gGASPs on acyclic graphs is in FPT with respect to the number of activities. For general graphs, we obtain a W[1]-hardness result, implying that this problem is unlikely to admit an FPT algorithm. This hardness result holds even for gGASPs on cliques; thus, it is also W[1]-hard to decide the existence of a Nash stable outcome in a standard GASP.

While we find that from an algorithmic point of view, individual stability is very similar to Nash stability, unfortunately, our FPT results do not extend to core stability: we prove that checking the existence of core stable assignments is NP-complete even for gGASPs on stars with two activities; for standard GASP, we can prove that this problem is hard if there are at least four activities. On the other hand, if there is only one activity, a core stable assignment always exists and can be constructed efficiently, for any network structure.

Another parameter we consider is the number of players. This restriction applies to many practical scenarios. For example, in research teams with limited human resources, there are a limited number of researchers who are able to conduct the projects. Somewhat surprisingly, we show that the parameterization by the number of players does not give

rise to an FPT algorithm for gGASPs on general networks. Specifically, for all stability notions we consider, it is W[1]-hard to decide the existence of a stable outcome even when the underlying graph is a clique. Again, our hardness result particularly implies the W[1]-hardness of computing stable outcomes in a standard GASP.

3.1 Model

An instance of the *Group Activity Selection Problem* (GASP) is given by a finite set of *players* $N = [n]$, a finite set of *activities* $A = A^* \cup \{a_\emptyset\}$, where $A^* = \{a_1, a_2, \ldots, a_p\}$ and $a_\emptyset$ is the *void activity*, and a *profile* $(\succeq_i)_{i \in N}$ of complete and transitive preference relations over the set of *alternatives* $(A^* \times [n]) \cup \{(a_\emptyset, 1)\}$. Intuitively, $a_\emptyset$ corresponds to staying alone and doing nothing; multiple agents can make that choice independently from each other.

We assume that we can determine in unit time whether each player i prefers (a, k) to (b, ℓ), prefers (b, ℓ) to (a, k), or is indifferent between them. We will write $x \succ_i y$ or $i : x \succ y$ to indicate that player i strictly prefers alternative x to alternative y; similarly, we will write $x \sim_i y$ or $i : x \sim y$ if i is indifferent between x and y. Also, given two sets of alternatives X, Y and a player i, we write $X \succ_i Y$ to indicate that i is indifferent among all alternatives in X as well as among all alternatives in Y, and prefers each alternative in X to each alternative in Y.

We say that two non-void activities a and b are *equivalent* if for every player $i \in N$ and every $\ell \in [n]$ it holds that $(a, \ell) \sim_i (b, \ell)$. A non-void activity $a \in A^*$ is called *copyable* if A^* contains at least n activities that are equivalent to a (including a itself). We say that player $i \in N$ *approves* an alternative (a, k) if $(a, k) \succ_i (a_\emptyset, 1)$.

An outcome of a GASP is an *assignment* of activities A to players N, i.e., a mapping $\pi : N \to A$. Given an assignment $\pi : N \to A$ and a non-void activity $a \in A^*$, we denote by $\pi^a = \{\, i \in N \mid \pi(i) = a \,\}$ the set of players assigned to a. Also, if $\pi(i) \neq a_\emptyset$, we denote by $\pi_i = \{i\} \cup \{\, j \in N \mid \pi(j) = \pi(i)\}$ the set of players assigned to the same activity as player $i \in N$; we set $\pi_i = \{i\}$ if $\pi(i) = a_\emptyset$. An assignment $\pi : N \to A$ for a GASP is *individually rational* (IR) if for every player $i \in N$ with $\pi(i) \neq a_\emptyset$ we have $(\pi(i), |\pi_i|) \succeq_i (a_\emptyset, 1)$. A coalition $S \subseteq N$ and an activity $a \in A^*$ *strongly block* an assignment $\pi : N \to A$ if $\pi^a \subseteq S$ and $(a, |S|) \succ_i (\pi(i), |\pi_i|)$ for all $i \in S$. An assignment $\pi : N \to A$ for a GASP is called *core stable* (CR) if it is individually rational, and there is no coalition $S \subseteq N$ and activity $a \in A^*$ such that S and a strongly block π. Given an assignment $\pi : N \to A$ of a gGASP, a player $i \in N$ is said to have

- an *NS-deviation* to activity $a \in A^*$ if i strictly prefers to join the group π^a, i.e., $(a, |\pi^a| + 1) \succ_i (\pi(i), |\pi_i|)$.
- an *IS-deviation* if it is an NS-deviation, and all players in π^a accept it, i.e., $(a, |\pi^a| + 1) \succeq_j (a, |\pi^a|)$ for all $j \in \pi^a$.

We now define a group activity selection problem where communication links between the players are represented by an undirected graph.

Definition 3.1. An instance of the *Group Activity Selection Problem with graph structure* (gGASP) is given by an instance $(N, (\succeq_i)_{i \in N}, A)$ of a GASP and a set of communication links between players $L \subseteq \{\, \{i, j\} \mid i, j \in N \wedge i \neq j \,\}$.

Again, a coalition $S \subseteq N$ is said to be *feasible* if S is connected in the graph (N, L). An outcome of a gGASP is a *feasible assignment* $\pi : N \to A$ such that π_i is a feasible coalition for every $i \in N$. We adapt the definitions of stability concepts to our setting as follows. We say that a deviation by a group of players is *feasible* if the deviating coalition itself is feasible; a deviation by an individual player where player i joins activity a is *feasible* if $\pi^a \cup \{i\}$ is feasible. We modify the definitions in the previous section by only requiring stability against feasible deviations. Note that an ordinary GASP (without graph structure) is equivalent to a gGASP where the underlying graph (N, L) is complete.

A key feature of gGASP as well as GASP is that players' preferences are *anonymous*, i.e., players do not care about the identities of the group members. We can thus show that checking whether a given feasible assignment is core stable is easy, irrespective of the structure of the social network. The proposition below generalizes Theorem 11 of [41]. Note that in many other contexts, deciding whether a given assignment is core stable is coNP-hard, for example in additively separable hedonic games [93].

Proposition 3.1. *Given an instance* $(N, (\succeq_i))_{i \in N}, A, L)$ *of* gGASP *and a feasible assignment* π *for that instance, we can decide in* $O(pn^3)$ *time whether* π *is core stable.*

Proof. Let $A = A^* \cup \{a_\emptyset\}$ and let $n = |N|$. By scanning the assignment π and the players' preferences, we can check whether π is individually rational. Now, suppose that this is the case. Then, for each $a \in A^*$ and each $s \in [n]$ we can check if there is a deviation by a connected coalition of size s that engages in a. To this end, we consider the set $S_{a,s}$ of all players who strictly prefer (a, s) to their assignment under π and verify whether $S_{a,s}$ has a connected component of size at least s that contains π^a; if this is the case, π^a (which is itself connected or empty) could be extended to a connected coalition of size exactly s that strongly blocks π. If no such deviation exists, π is core stable. The existence of a

connected component of a given size can be checked in $O(n^2)$ time by using depth first search algorithm. □

In what follows, we will be especially interested in gGASPs where (N, L) is *acyclic*. This restriction guarantees the existence of stable outcomes in many other cooperative game settings [46] (see also Chapter 2). However, this is not the case for gGASP: here, all stable outcomes under consideration may fail to exist.

Example 3.1 (Stalker game)**.** Consider a gGASP with $N = \{1, 2\}$, $A^* = \{a\}$, $L = \{\{1, 2\}\}$, where preferences $(\succeq_i)_{i \in N}$ are given as follows:

$$1 : (a, 1) \succ (a_\emptyset, 1)$$
$$2 : (a, 2) \succ (a_\emptyset, 1)$$

Thus, player 1 wishes to participate in activity a alone, while player 2 (the "stalker") wants to participate in activity a together with player 1.

This instance admits no Nash stable outcomes: If all players engage in the void activity, player 1 wants to start doing activity a. If player 1 does activity a, then player 2 wants to join her coalition, causing player 1 to deviate to the void activity. □

Similarly, a core stable outcome is not guaranteed to exist even for gGASPs on paths and stars, as the following example shows.

Example 3.2. Consider a gGASP with $N = \{1, 2, 3\}$, $A^* = \{a, b\}$, $L = \{\{1, 2\}, \{2, 3\}\}$, where preferences $(\succeq_i)_{i \in N}$ are given as follows:

$$1 : (b, 2) \succ (a, 3) \succ (a_\emptyset, 1)$$
$$2 : (a, 2) \succ (b, 2) \succ (a, 3) \succ (a_\emptyset, 1)$$
$$3 : (a, 3) \succ (b, 1) \succ (a, 2) \succ (a_\emptyset, 1)$$

We will argue that each individually rational feasible assignment π admits a strongly blocking feasible coalition and activity. If all players do nothing, then player 3 and activity b strongly block π. Now, there are only four individually rational feasible assignments where some player is engaged in a non-void activity; each of them is strongly blocked by some coalition and activity as follows (we write $S \to x$ to indicate that coalition S strongly blocks π together with activity x):

(1) $\pi(1) = b$, $\pi(2) = b$, $\pi(3) = a_\emptyset$: $\{2, 3\} \to a$;

(2) $\pi(1) = a_\emptyset$, $\pi(2) = a$, $\pi(3) = a$: $\{3\} \to b$;

(3) $\pi(1) = a_\emptyset$, $\pi(2) = a_\emptyset$, $\pi(3) = b$: $\{1, 2, 3\} \to a$;

(4) $\pi(1) = a$, $\pi(2) = a$, $\pi(3) = a$: $\{1, 2\} \to b$; □

We showed that in the context of hedonic games, acyclicity is sufficient for individually stable outcomes to exist: an individually stable partition of players always exists and can be computed in polynomial time. In contrast, it turns out that for gGASPs this is not the case: an individually stable outcome may fail to exist even if the underlying social network is a path; moreover, this may happen even if there are only three players and their preferences are strict.

Example 3.3. Consider a gGASP with $N = \{1, 2, 3\}$, $A^* = \{a, b, c\}$, $L = \{\{1, 2\}, \{2, 3\}\}$, where players' preferences are as follows:

$$1 : (b, 2) \succ (a, 1) \succ (c, 3) \succ (c, 2) \succ (c, 1) \succ (a_\emptyset, 1)$$
$$2 : (c, 3) \succ (c, 2) \succ (a, 2) \succ (b, 2) \succ (b, 1) \succ (a_\emptyset, 1)$$
$$3 : (c, 3) \succ (a, 2) \succ (a, 1) \succ (a_\emptyset, 1)$$

We will argue that each individually rational feasible assignment π admits an IS-deviation. Indeed, if $\pi(1) = a_\emptyset$ then no player is engaged in c and hence player 1 can deviate to c. Similarly, if $\pi(2) = a_\emptyset$ then no player is engaged in b and hence player 2 can deviate to b. There are 9 individually rational feasible assignments where $\pi(1) \neq a_\emptyset$, $\pi(2) \neq a_\emptyset$; for each of them we can find an IS deviation as follows (we write $i \to x$ to indicate that player i has an IS-deviation to activity x):

(1) $\pi(1) = a$, $\pi(2) = b$, $\pi(3) = a_\emptyset$: $1 \to b$;

(2) $\pi(1) = b$, $\pi(2) = b$, $\pi(3) = a_\emptyset$: $3 \to a$;

(3) $\pi(1) = b$, $\pi(2) = b$, $\pi(3) = a$: $2 \to a$;

(4) $\pi(1) = c$, $\pi(2) = a$, $\pi(3) = a$: $2 \to c$;

(5) $\pi(1) = c$, $\pi(2) = b$, $\pi(3) = a_\emptyset$: $3 \to a$;

(6) $\pi(1) = c$, $\pi(2) = b$, $\pi(3) = a$: $2 \to a$;

(7) $\pi(1) = c$, $\pi(2) = c$, $\pi(3) = a_\emptyset$: $3 \to a$;

(8) $\pi(1) = c$, $\pi(2) = c$, $\pi(3) = a$: $3 \to c$;

(9) $\pi(1) = c$, $\pi(2) = c$, $\pi(3) = c$: $1 \to a$.

Notice that the instance does not admit a core stable outcome either: if such an outcome existed, the assignment would satisfy individual stability due to the fact that all the preferences are strict, a contradiction to what we have seen above. □

3.2 Copyable Cases

If all activities are copyable, we can effectively treat gGASP as a non-transferable utility game on a graph. In particular, we can invoke the result of [46] concerning the stability of non-transferable utility games on trees. Thus, requiring all activities to be copyable allows us to circumvent the non-existence result for the core (Example 3.2). The argument is constructive.

Theorem 3.1 (implicit in the work of [46])**.** *For every* gGASP *where each activity* $a \in A^*$ *is copyable and* (N, L) *is a forest, a core stable feasible assignment exists and can be found in time polynomial in* p *and* n*.*

Proof. If the input graph (N, L) is a forest, we can process each of its connected components separately, so we assume that (N, L) is a tree. Prior to giving a formal description of the algorithm (Algorithm 4), we outline the basic idea. We choose an arbitrary node r as the root and construct a rooted tree (N, T) by orienting the edges in L towards the leaves. We denote by $(N, T|S)$ the subdigraph induced by $S \subseteq N$, i.e., $T|S = \{\, (i, j) \in T \mid i, j \in S \,\}$. We define $\mathcal{B}(i)$ to be the set of connected subsets of $\mathrm{desc}(i)$ including i for each $i \in N$.

The algorithm has two different phases: the bottom-up and the top-down phase.

- *Bottom-up phase*: In the bottom-up phase, we will determine a *guaranteed* activity $a(i)$ and coalition $S(i)$ for every subroot i. To this end, we choose a connected subset $S(i)$ of $\mathrm{desc}(i)$ that maximizes i's utility under the constraint that every descendant j of i in the coalition can agree, i.e., for any descendant $j \in S(i)$ of i, $(a(i), |S(i)|)$ is at least as preferred as $(a(j), |S(j)|)$. The utility level of $(a(i), |S(i)|)$ determined for each player $i \in N$ can be interpreted as i's guaranteed utility in the final outcome.

- *Top-down phase*: In the top-down phase, the algorithm builds a feasible assignment π, by iteratively choosing a root r' of the remaining rooted trees and reassigning the activity $a(r')$ to its coalition $S(r')$. Since each activity is copyable, we can always find an activity that is equivalent to $a(r')$ and has not been used by their predecessors.

This procedure is formalized in Algorithm 4.

We will now argue that Algorithm 4 correctly finds a core stable feasible assignment. Observe that the following lemma holds due to the **if** statement in line 7 of the algorithm.

Algorithm 4: Finding core stable partitions

input : tree (N, L), activity set $A = A^* \cup \{a_\emptyset\}$, $r \in N$, preference $\succeq_i$, $i \in N$
output : $\pi : N \to A$

// Bottom-up phase: assign activities to players in a bottom-up manner;
make a rooted tree (N, T) with root r by orienting all the edges in L;
initialize $S(i) \leftarrow \{i\}$ and $a(i) \leftarrow a_\emptyset$ for each $i \in N$;
foreach $t = 0, \ldots,$ height(r) **do**
    foreach $i \in N$ *with* height$(i) = t$ **do**
        // Find i's favourite alternative where all members of i's coalition can agree to join. **foreach** $(a, k) \in (A^* \times [n]) \cup \{(a_\emptyset, 1)\}$ **do**
            if *there exists* $S \in \mathcal{B}(i)$ *such that* $|S| = k$, $(a, k) \succeq_j (a(j), |S(j)|)$ *for all* $j \in S$, *and* $(a, k) \succ (a(i), |S(i)|)$ **then**
                set $S(i) \leftarrow S$ and $a(i) \leftarrow a$;
// Top-down phase: relabel players with their predecessor's activities;
set $N' \leftarrow N$ and $A' \leftarrow A^*$;
while $N' \neq \emptyset$ **do**
    choose a root r' of some connected component of the digraph $(N', T|N')$ and find an activity $b \in A' \cup \{a_\emptyset\}$ that is equivalent to $a(r')$;
    set $\pi(i) \leftarrow b$ for all $i \in S(r')$;
    set $N' \leftarrow N' \setminus S(r')$ and $A' \leftarrow A' \setminus \{b\}$;

Lemma 3.1. *For all $i \in N$, the following statements hold:*

(i) *i has no incentive to deviate to an alternative of size 1, i.e., $(a(i), |S(i)|) \succeq_i (b, 1)$ for all $b \in A$,*

(ii) *all players j in $S(i)$ weakly prefer $(a(i), |S(i)|)$ to their guaranteed alternative $(a(j), |S(j)|)$.*

Now, by (ii) in Lemma 3.1, it can be easily verified that

$$(\pi(i), |\pi_i|) \succeq_i (a(i), |S(i)|) \text{ for all } i \in N. \tag{3.1}$$

Combining this with (i), we know that at the assignment π, all players weakly prefer their alternatives to engaging alone in unused activities or the void activity. It thus remains to show that no connected coalition together with a non-void activity strongly blocks π.

Take any connected subset $S \subseteq N$ and activity $a \in A^*$. Let i be the subroot of the coalition S so that $S \in \mathcal{B}(i)$. First, consider the case when $(a(i), |S(i)|) \succeq_i (a, |S|)$. By (3.1), it is clear that the coalition S and the activity a do not strongly block π. Second, consider the case when $(a, |S|) \succ_i (a(i), |S(i)|)$. By line 7 of the algorithm, this means that there is a player $j \in S$ such that $(a(j), |S(j)|) \succ_j (a, |S|)$. Thus, S and a do not strongly block π. We conclude that π is core stable.

It remains to analyze the running time of Algorithm 4. Consider the execution of the algorithm for a fixed player i. Let $d = |\text{desc}(i)|$. Line 7 checks whether there is a connected coalition of size k that can engage in a. Similarly to the proof of Proposition 3.1, we do this by computing the set S of all descendants in $\text{desc}(i) \setminus \{i\}$ who weakly prefer (a, k) to their guaranteed coalition and verify whether the set $S \cup \{i\}$ has a connected component of size at least k. This procedure requires at most d queries: no descendant of i is queried more than once. Summing over all players, we conclude that the number of queries for the bottom-up phase is bounded by $O(n^3 p)$. It is immediate that the top-down phase can be done in polynomial time. This completes the proof of the theorem. □

Similarly, an individually stable outcome is guaranteed to exist in copyable cases if the underlying graph is acyclic. Moreover, we can adapt the result for hedonic games and obtain an efficient algorithm for computing an individually stable outcome. The proof can be found in Appendix A.2.

Theorem 3.2. *Each instance of* gGASP *where each activity $a \in A^*$ is copyable and (N, L) is a forest admits an individually stable feasible assignment; moreover, such an assignment can be found in time polynomial in p and n.*

The stalker game in Example 3.1 does not admit a Nash stable outcome even if we make all activities copyable. Thus, in contrast to core and individual stability, there is no existence guarantee for Nash stability even if activities are copyable. However, with copyable activities, we can still *check* for the existence of a Nash stable outcome in polynomial time if the social network is acyclic.

Theorem 3.3. *Given an instance* $(N, A, (\succeq_i)_{i\in N}, L)$ *of* gGASP *where each activity* $a \in A^*$ *is copyable and the graph* (N, L) *is a forest, one can decide whether it admits a Nash stable outcome in time polynomial in* p *and* n*.*

Proof. Again, we assume that (N, L) is a tree. We choose an arbitrary node as the root and construct a rooted tree by orienting the edges in L towards the leaves. We denote by $\mathrm{ch}(i) = \{j_1, j_2, \ldots, j_{|\mathrm{ch}(i)|}\}$ the set of children of i; and we denote by $\mathrm{desc}(i, c)$ the set of the descendants of the first c-th children of i (including $j_1, j_2, \ldots, j_c$) in the rooted tree. Then, for each player $i \in N$, each $c \in [|\mathrm{ch}(i)|]$, each alternative $(a, k) \in X$, and $t \in [k]$ we set $\mathsf{NS}[i, c, (a, k), t]$ to *true* if there exists a feasible assignment $\pi : N \to A$ such that $|\pi_i \cap (\mathrm{desc}(i, c) \cup \{i\})| = t$, $\pi(i) = a$, each player in $\pi_i \cap (\mathrm{desc}(i, c) \cup \{i\})$ likes (a, k) at least as much as any alternative she can deviate to (including the void activity), and no player in $\mathrm{desc}(i, c) \setminus \pi_i$ has an NS feasible deviation. Otherwise, we set $\mathsf{NS}[i, c, (a, k), t]$ to *false*. By construction, there exists a Nash stable feasible assignment if and only if $\mathsf{NS}[r, |\mathrm{ch}(r)|, (a, k), k]$ is *true* for some alternative $(a, k) \in X$, where r is the root of the rooted tree.

For each player $i \in N$, each $c \in [|\mathrm{ch}(i)|]$, each alternative $(a, k) \in X$, and each $t \in [k]$, we initialize $\mathsf{NS}[i, c, (a, k), t]$ to *true* if $t = 1$ and i weakly prefers (a, k) to any alternative of size 1, and we set $\mathsf{NS}[i, c, (a, k), t]$ to *false* otherwise. Then, for $i \in N$ from the bottom to the root, we iterate through all the children j_c of i and update $\mathsf{NS}[i, c, (a, k), t]$ step-by-step; more precisely, for each child j_c of i and for $t \in [k]$, we set $\mathsf{NS}[i, c, (a, k), t]$ to *true* if

- $t \geq 2$ and there exists an $x \in [t - 1]$ such that both $\mathsf{NS}[i, c - 1, (a, k), t - x]$ and $\mathsf{NS}[j, c, (a, k), x]$ are *true*, or
- $\mathsf{NS}[i, c - 1, (a, k), t]$ is *true*, and players i and j_c can be *separated* from each other, i.e., there exists $(b, \ell) \in X$ such that (i) $\mathsf{NS}[j_c, |\mathrm{ch}(j)|, (b, \ell), \ell]$ is *true*, (ii) $b = a_\emptyset$ or $(a, k) \succeq_i (b, \ell + 1)$, and (iii) $a = a_\emptyset$ or $(b, \ell) \succeq_j (a, k + 1)$.

In cases where $\mathsf{NS}[r, |\mathrm{ch}(r)|, (a, k), k]$ is *true* for some alternative $(a, k) \in X$, a Nash stable feasible assignment can be found using dynamic programming.

This can be done in polynomial time since the size of the dynamic programming table is at most n^3p and each entry can be filled in time $O(n^2p)$. This completes the proof of the theorem. □

We note that these tractability results for copyable cases do not extend to arbitrary graphs: Ballester [15] showed that it is NP-complete to determine the existence of core or individually or Nash stable outcomes for anonymous hedonic games, which can be considered as a subclass of gGASPs whose graph is a clique.

3.3 NP-completeness Results

We now move on to the case where each activity can be used at most once. For other types of cooperative games, many desirable outcomes can be computed in polynomial time if the underlying network structure is simple [36, 52]. In particular, we showed it is easy to compute stable partitions for hedonic games on trees with polynomially bounded number of connected coalitions. However, computing stable solutions of gGASP turns out to be NP-complete even if the underlying network is a path, a star, or a graph with constant-size connected components. For each family of graphs, we will reduce from a different combinatorial problem that is structurally similar to our problem.

Paths. Our proof for paths is by reduction from a restricted version of the NP-complete problem RAINBOW MATCHING. Given a graph G and a set of colors $\mathcal{C}$, a *proper edge coloring* is a mapping $\phi : E \to \mathcal{C}$ where $\phi(e) \neq \phi(e')$ for all edges e, e' such that $e \neq e'$ and $e \cap e' \neq \emptyset$. Without loss of generality, we assume that ϕ is surjective. A *properly edge-colored graph* $(G, \mathcal{C}, \phi)$ is a graph together with a set of colors and a proper edge coloring. A matching M in an edge-colored graph $(G, \mathcal{C}, \phi)$ is called a *rainbow matching* if all edges of M have different colors. Given a properly edge-colored graph $(G, \mathcal{C}, \phi)$ together with an integer k, RAINBOW MATCHING asks whether G admits a rainbow matching with at least k edges. Le and Pfender [69] show that RAINBOW MATCHING remains NP-complete even for properly edge-colored paths.

Theorem 3.4. *Given an instance of* gGASP *whose underlying graph is a path, it is* NP*-complete to determine whether it has a Nash stable feasible assignment.*

Proof. Clearly, our problems are contained in NP since we can easily check whether a given assignment is Nash stable. The hardness proof proceeds by a reduction from PATH RAINBOW MATCHING.

Construction. Given an instance $(G, \mathcal{C}, \phi, k)$ of PATH RAINBOW MATCHING where $|\mathcal{C}| = q$, we create a vertex player v for each $v \in V$ and an edge player e for each $e \in E$. To create the social network, we start with G and place each edge player in the middle of the respective edge, i.e., we let $N_G = V \cup E$ and $L_G = \{\, \{v, e\} \mid v \in e \in E \,\}$. To the right of the graph (N_G, L_G), we attach a path consisting of "garbage collectors" $\{g_1, g_2, \ldots, g_{q-k}\}$ and q copies (N_c, L_c) of the stalker game where $N_c = \{c_1, c_2\}$ and $L_c = \{\{c_1, c_2\}\}$ for each $c \in \mathcal{C}$. See Figure 3.1.

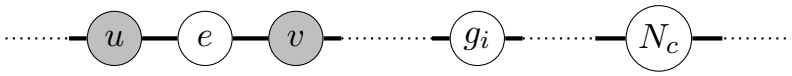


Figure 3.1: The graph constructed in the hardness proof for gGASPs on paths.

We introduce a color activity c for each color $c \in \mathcal{C}$. Each vertex player v approves color activities $\phi(e)$ of its adjacent edges e with size 3; each edge player e approves the color activity $\phi(e)$ of its color with size 3; each garbage collector g_i approves any color activity c with size 1; finally, for players in N_c, $c \in \mathcal{C}$, player c_1 approves its color activity c with size 1, whereas player c_2 approves c with size 2.

Correctness. We show that G has a rainbow matching of size at least k if and only if there exists a Nash stable feasible assignment.

Suppose that there exists a rainbow matching M of size k. We construct a feasible assignment π where for each $e = \{u, v\} \in M$ we set $\pi(e) = \pi(u) = \pi(v) = \phi(e)$, each garbage collector g_i, $i \in [q - k]$, is arbitrarily assigned to one of the remaining $q - k$ color activities, and the remaining players are assigned to the void activity. The assignment π is Nash stable, since every garbage collector as well as every edge or vertex player assigned to a color activity are allocated their top alternative, and no remaining player has an NS feasible deviation.

Conversely, suppose that there is a Nash stable feasible assignment π. Let $M = \{\, e \in E(G) \mid \pi(e) \in \mathcal{C} \,\}$. We will show that M is a rainbow matching of size at least k. To see this, notice that π cannot allocate a color activity to a member of N_c, since otherwise no feasible assignment would be Nash stable. Further, at most $q - k$ color activities are allocated to the garbage collectors, which means that at least k color activities should be assigned to vertex and edge players. The only individually rational way to do this is to select triples of the form (u, e, v) where $e = \{u, v\} \in E(G)$ and assign to them their color activity $\phi(e)$. Thus, M is a rainbow matching of size at least k. □

Stars. For gGASPs on stars we provide a reduction from the NP-complete problem MINIMUM MAXIMAL MATCHING (MMM). Given a graph G and a positive integer $k \leq |E|$,

MMM asks whether G admits a maximal matching with at most k edges. The problem remains NP-complete for bipartite graphs [47].

Theorem 3.5. *Given an instance of* gGASP *whose underlying graph is a star, it is* NP*-complete to determine whether it has a Nash stable feasible assignment.*

Proof. Clearly, our problem is in NP. To prove NP-hardness, we reduce from MMM on bipartite graphs.

Construction. Given a bipartite graph $G = (U \cup V, E)$ with vertex bipartition (U, V) and an integer k, we create a star with center c and $|V| + 1$ leaves: one leaf for each vertex player $v \in V$ plus one stalker s. See Figure 3.2.

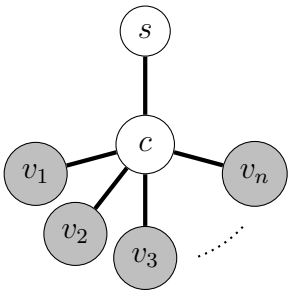


Figure 3.2: The graph constructed in the hardness proof for gGASPs on stars.

We introduce an activity u for each $u \in U$, and two additional activities a and b. A player $v \in V$ approves $(u, 1)$ for each activity u such that $\{u, v\} \in E$ as well as $(a, |V| - k + 1)$ and prefers the former to the latter. That is, $(u, 1) \succ_v (a, |V| - k + 1)$ for every $u \in U$ with $\{u, v\} \in E$; v is indifferent among the activities associated with its neighbors in the graph, that is, $(u, 1) \sim_v (u', 1)$ for all $u, u' \in U$ such that $\{u, v\} \in E$ and $\{u', v\} \in E$. The center player c approves both $(a, |V| - k + 1)$ and $(b, 1)$, and prefers the former to the latter, i.e., $(a, |V| - k + 1) \succ_c (b, 1) \succ_c (a_\emptyset, 1)$. Finally, the stalker s only approves $(b, 2)$.

Correctness. We now show that G admits a maximal matching M with at most k edges if and only if our instance of gGASP admits a Nash stable assignment.

Suppose that G admits a maximal matching M with at most k edges. We construct a feasible assignment π by setting $\pi(v) = u$ for each $\{u, v\} \in M$ with $u \in U$ and $v \in V$, assigning the other $|V| - k$ vertex players $v \in V$ and the center to a, and assigning the stalker s to the void activity. Clearly, the center c has no incentive to deviate and no vertex player in a singleton coalition wants to deviate to the coalition of the center. Further, no vertex $v \in V$ with $\pi(v) = a$ has an NS-deviation to an unused activity $u \in U$, since if π admits such a deviation, this would mean that $M \cup \{u, v\}$ forms a matching, contradicting maximality of M. Finally, the stalker player does not deviate since the center does not engage in b. Hence, π is Nash stable.

Conversely, suppose that there exists a Nash stable feasible assignment π and let $M = \{ \{\pi(v), v\} \mid v \in V \wedge \pi(v) \in U \}$. We will show that M is a maximal matching of size at most k. By Nash stability, the stalker player should not have an incentive to deviate, and hence the center player and $|V| - k$ vertex players are assigned to activity a. It follows that k vertex players are not assigned to a, and therefore $|M| \leq k$. Moreover, M is a matching since each vertex player is assigned to at most one activity, and by individual rationality each activity can be assigned to at most one player. Now suppose towards a contradiction that M is not maximal, i.e., there exists an edge $\{u, v\} \in E$ such that $u \in U$, $v \in V$, and $M \cup \{u, v\}$ is a matching. This would mean that in π no player is assigned to u, and v is assigned to the activity a; hence, v has an NS-deviation to u, contradicting the Nash stability of π. □

Small Components. In the analysis of cooperative games on social networks one can usually assume that the social network is connected: if this is not the case, each connected component can be processed separately. This is also the case for gGASP as long as all activities are copyable. However, if each activity can only be used by a single group, different connected components are no longer independent, as they have to choose from the same pool of activities. Indeed, we will now show that the problem of finding stable outcomes remains NP-hard even if the size of each connected component is at most four. Our hardness proof for this problem proceeds by reduction from a restricted version of 3SAT. Specifically, we consider (3,B2)-SAT: in this version of 3SAT each clause contains exactly 3 literals, and each variable occurs exactly twice positively and twice negatively. This problem is known to be NP-complete [19].

Theorem 3.6. *Given an instance of* gGASP *whose underlying graph has connected components whose size is bounded by* 4*, it is* NP*-complete to determine whether it has a Nash stable feasible assignment.*

Proof. Our problem is in NP. We reduce from (3,B2)-SAT.

Construction. Consider a formula ϕ with variable set X and clause set C, where for each variable $x \in X$ we write x_1 and x_2 for the two positive occurrences of x, and $\bar{x}_1$ and $\bar{x}_2$ for the two negative occurrences of x. For each $x \in X$, we introduce four players $p_1(x), p_2(x), \bar{p}_1(x)$, and $\bar{p}_2(x)$. For each clause $c \in C$, we introduce one stalker s_c and three other players c_1, c_2, and c_3. The network (N, L) consists of one component for each clause—a star with center s_c and leaves c_1, c_2, and c_3—and of two components for each variable $x \in X$ consisting of a single edge each: $\{p_1(x), p_2(x)\}$ and $\{\bar{p}_1(x), \bar{p}_2(x)\}$. See Figure 3.3. Thus, the size of each component of this graph is at most 4.

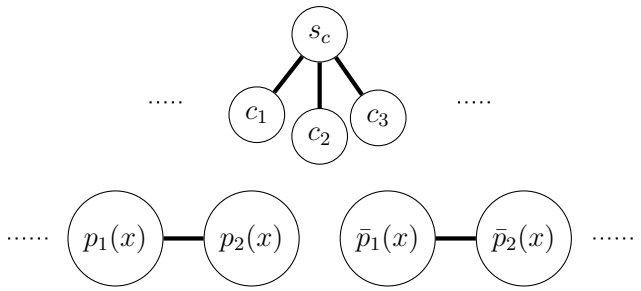


Figure 3.3: The graph used in the hardness proof for gGASPs on graphs with small components.

For each $x \in X$ we introduce one variable activity x, two positive literal activities x_1 and x_2, and two negative literal activities $\bar{x}_1$ and $\bar{x}_2$, which correspond to the four occurrences of x; also, we introduce two further activities $a(x)$ and $\bar{a}(x)$. Finally, we introduce an activity c for each clause $c \in C$. Thus,

$$A^* = \bigcup_{x \in X} \{x, x_1, x_2, \bar{x}_1, \bar{x}_2, a(x), \bar{a}(x)\} \cup C.$$

For each $x \in X$ the preferences of the positive literal players $p_1(x)$ and $p_2(x)$ are given as follows:

$$p_1(x)\colon (x,2) \succ (x,1) \succ (x_1,1) \succ (x_2,2) \succ (a(x),1) \succ (a_\emptyset,1),$$
$$p_2(x)\colon (x,2) \succ (x_2,1) \succ (x_1,2) \succ (a(x),2) \succ (a_\emptyset,1).$$

If one of the positive literal players $p_1(x)$ and $p_2(x)$ is engaged in the void activity and the other is engaged alone in a non-void activity, this would cause the former player to deviate to another activity; thus, in a Nash stable assignment, none of the activities $a(x)$ and $a_\emptyset$ can be assigned to positive literal players. Similarly, for each $x \in X$ the preferences of the negative literal players $\bar{p}_1(x)$ and $\bar{p}_2(x)$ are given as follows:

$$\bar{p}_1(x)\colon (x,2) \succ (x,1) \succ (\bar{x}_1,1) \succ (\bar{x}_2,2) \succ (\bar{a}(x),1) \succ (a_\emptyset,1),$$
$$\bar{p}_2(x)\colon (x,2) \succ (\bar{x}_2,1) \succ (\bar{x}_1,2) \succ (\bar{a}(x),2) \succ (a_\emptyset,1).$$

As argued above, Nash stable assignments cannot allocate activities $\bar{a}(x)$ and $a_\emptyset$ to negative literal players. Hence, if there exists a Nash stable assignment, there are only two possible cases:

- both players $p_1(x)$ and $p_2(x)$ are assigned to x, and players $\bar{p}_1(x)$ and $\bar{p}_2(x)$ are assigned to $\bar{x}_1$ and $\bar{x}_2$, respectively;
- both players $\bar{p}_1(x)$ and $\bar{p}_2(x)$ are assigned to x, and players $p_1(x)$ and $p_2(x)$ are assigned to x_1 and x_2, respectively.

For players in N_c where ℓ_1^c, ℓ_2^c, and ℓ_3^c are the literals in a clause c, the preferences are given by

$$c_r : (\ell_r^c, 1) \succ (c, 2) \succ (a_\emptyset, 1), \qquad (r = 1, 2, 3)$$
$$s_c : (\ell_1^c, 2) \sim (\ell_2^c, 2) \sim (\ell_3^c, 2) \sim (c, 2) \succ (a_\emptyset, 1).$$

That is, players c_1, c_2, and c_3 prefer to engage alone in their approved literal activity, whereas s_c wants to join one of the adjacent leaves whenever $\pi(s_c) = a_\emptyset$ and that leaf is assigned a literal activity; however, the leaf would then prefer to switch to the void activity. This means that if there exists a Nash stable outcome, at least one of the literal activities must be used outside of N_c, and some leaf and the stalker s_c must be assigned to activity c.

Correctness. We will show that ϕ is satisfiable if and only if there exists a Nash stable outcome.

Suppose that there exists a truth assignment that satisfies ϕ. First, for each variable x that is set to *true*, we assign positive literal activities x_1 and x_2 to the positive literal players $p_1(x)$ and $p_2(x)$, respectively, and assign x to the negative literal players $\bar{p}_1(x)$ and $\bar{p}_2(x)$. For each variable x that is set to *false*, we assign negative literal activities $\bar{x}_1$ and $\bar{x}_2$ to the negative literal players $\bar{p}_1(x)$ and $\bar{p}_2(x)$, respectively, and assign x to the positive literal players $p_1(x)$ and $p_2(x)$. Note that this procedure uses at least one of the literal activities ℓ_1^c, ℓ_2^c and ℓ_3^c of each clause $c \in C$, since the given truth assignment satisfies ϕ. Then, for each clause $c \in C$, we select a player c_j whose approved activity ℓ_j^c has been assigned to some literal player, and assign c_j and the stalker to c, and the rest of the clause players to their approved literal activity if it is not used yet, and to the void activity otherwise. It is easy to see that the resulting assignment π is Nash stable.

Conversely, suppose that there exists a Nash stable feasible assignment π. By Nash stability, for each variable $x \in X$, either a pair of positive literal players $p_1(x)$ and $p_2(x)$ or a pair of negative literal players $\bar{p}_1(x)$ and $\bar{p}_2(x)$ should be assigned to the corresponding pair of literal activities; in addition, for each clause $c \in C$, the stalker s_c and one of the players c_1, c_2, and c_3 should engage in the activity c, thereby implying that the approved literal activity of the respective leaf should be assigned to some literal players. Then, take the truth assignment that sets the variable x to *true* if its positive literal players $p_1(x)$ and $p_2(x)$ are assigned to positive literal activities x_1 and x_2; otherwise, x is set to *false*. This assignment can be easily seen to satisfy ϕ. □

The hardness reductions for the core and individual stability are similar to the respective reductions for Nash stability; essentially, we have to replace copies of the stalker game from Example 3.1 with copies of games with an empty core (Example 3.2) or with no individually stable outcome (Example 3.3).

Theorem 3.7. *Given an instance of* gGASP *whose underlying graph is a path, a star, or has connected components whose size is bounded by* 3*, it is* NP*-complete to determine whether it has a core stable feasible assignment, and it is* NP*-complete to determine whether it has an individually stable feasible assignment.*

Proof. Clearly, our problems are contained in NP for any social network due to Proposition 3.1 and the fact that we can easily check whether a given assignment is individually stable. The hardness proofs can be found in Appendix A.2. □

3.4 Few Activities

In the instances of gGASP that are created in our hardness proofs, the number of activities p is unbounded. In reality, however, there are many settings where this parameter can be very small. For instance, when organizing social events for a workshop, the number of activities one could organize is often restricted, due to a limited number of facilities for sports competition or a limited number of buses for a bus trip. It is thus natural to wonder what can be said when there are few activities to be assigned. It turns out that for each of the restricted families of graphs considered in the previous section, finding some stable assignments in gGASP is fixed parameter tractable with respect to the number of activities. The basic idea behind each of the algorithms we present is that we fix a set of activities that will be assigned to the players, and for each possible subset $B \subseteq A^*$ of activities we check whether there exists a stable assignment using the activities from that subset only.

3.4.1 Small Components

We first present an algorithm for small components based on dynamic programming, allowing us to build up the set B step-by-step. We order the components, and, for each prefix of that ordering, we check if a given subset of activities can be assigned to that prefix in a stable way. Within each component, we have enough time to consider all possible assignments, and each potential deviation involves at most one component. The resulting algorithm is FPT with respect to the combined parameter $p + c$, where c is a bound on the size of the components of the network.

Theorem 3.8. *There exists an algorithm that given an instance of* gGASP *checks whether it has a Nash stable or an individually stable feasible assignment, finds one if it exists, and runs in time* $O(8^p p^{c+1} c^2 n)$ *where* p *is the number of activities and* c *is the maximum size of connected components.*

Proof. We give a dynamic programming algorithm. Suppose our graph (N, L) has k connected components $(N_1, L_1), (N_2, L_2), \ldots, (N_k, L_k)$ and each component has size at most c, i.e., $|N_i| \leq c$ for all $i \in [k]$. For each $i \in [k]$, each set $B \subseteq A^*$ of activities assigned to N, and each set $B' \subseteq B$ of activities assigned to $\bigcup_{j=1}^{i} N_j$, we let $\mathsf{NS}[i, B, B']$ and $\mathsf{IS}[i, B, B']$ denote whether there is such an assignment that gives rise to a Nash stable outcome and an individually stable outcome, respectively. Specifically, $\mathsf{NS}[i, B, B']$ (respectively, $\mathsf{IS}[i, B, B']$) is *true* if there exists an individually rational feasible assignment $\pi : \bigcup_{j=1}^{i} N_j \to A$ such that

- the set of activities assigned to $\bigcup_{j=1}^{i} N_j$ is exactly B'; and
- no player in $\bigcup_{j=1}^{i} N_j$ has an NS (respectively, IS) feasible deviation to an activity in B' or to an activity in $A^* \setminus B$.

Otherwise, $\mathsf{NS}[i, B, B']$ (respectively, $\mathsf{IS}[i, B, B']$) is *false*.

For $i = 1$, each $B \subseteq A^*$, and each $B' \subseteq B$, we compute the value of $\mathsf{NS}[1, B, B']$ (respectively, $\mathsf{IS}[1, B, B']$) by trying all possible mappings $\pi : N_1 \to B' \cup \{a_\emptyset\}$, and checking whether it is an individually rational feasible assignment using all activities in B' and such that no player in N_1 has an NS (respectively, IS) feasible deviation to a used activity in B' or an unused activity in $A^* \setminus B$.

For $i = 2, 3, \ldots, k$, each $B \subseteq A^*$, and $B' \subseteq B$, we set $\mathsf{NS}[i, B, B']$ (respectively, $\mathsf{IS}[i, B, B']$) to *true* if there exists a bipartition of B' into P and Q such that $\mathsf{NS}[i-1, B, P]$ is *true* and there exists a mapping $\pi : N_i \to Q \cup \{a_\emptyset\}$ such that π is an individually rational feasible assignment using all the activities in Q, and no player in N_i has an NS (respectively, IS) feasible deviation to a used activity in Q or an unused activity in $A^* \setminus B$. This property holds because players cannot deviate to an activity played within a different connected component by feasibility.

It is not difficult to see that a Nash stable (respectively, individually stable) solution exists if and only if $\mathsf{NS}[k, B, B]$ (respectively, $\mathsf{IS}[k, B, B]$) is *true* for some $B \subseteq A^*$. If this is the case, such a stable feasible assignment can be found using standard dynamic programming techniques. The size of the dynamic programming table $\mathsf{NS}[i, B, B']$ (respectively, $\mathsf{IS}[i, B, B']$) is at most $4^p n$. For each entry, we consider $O(2^p)$ bipartitions of B' and the time required to find a stable assignment for a single component is $O(p^{c+1}c^2)$. Thus, each entry can be filled in time $O(2^p p^{c+1} c^2)$. This completes the proof. □

The FPT result for graphs with small connected components can also be adapted to the core. The proof can be found in Appendix A.2.

Theorem 3.9. *There exists an algorithm that given an instance of* gGASP *checks whether it has a core stable feasible assignment, finds one if it exists, and runs in time* $O(8^p p^{c+1} c^3 n)$ *where* p *is the number of activities and* c *is the maximum size of connected components.*

3.4.2 Acyclic Graphs

We will next show that computing Nash and individually stable outcomes is in FPT for arbitrary acyclic networks. Essentially, we will construct a rooted tree and check in a bottom-up manner whether there is a partial assignment that is extensible to a stable outcome. We note that this FPT result has been generalized to graphs with bounded tree-width by Gupta et al. [58], based on similar dynamic programming techniques.

Theorem 3.10. *The problem of deciding whether an instance of* gGASP *with* $|A^*| = p$ *whose underlying social network* (N, L) *is a forest has a Nash stable feasible assignment is in* FPT *with respect to* p*.*

Proof. We will first present a proof for the case where (N, L) is a tree; in the end, we will show how to extend the result to arbitrary forests. Fix an instance $(N, A, (\succeq_i)_{i \in N}, L)$ of gGASP such that (N, L) is a tree. We choose an arbitrary node in N as the root of this tree, thereby making (N, L) a rooted tree. Intuitively, we fix the set $B \subseteq A^*$ of activities assigned to the set of all players; then, we process the nodes from the leaves to the root, and for each player i and each subset B' of B, we check whether there is a partial assignment of B' to the descendants of i that is extensible to a stable assignment.

Formally, for each $i \in N$, each $B \subseteq A^*$, each $B' \subseteq B$, each $(a, k) \in (B' \times [n]) \cup \{(a_\emptyset, 1)\}$, and each $t \in [k]$, we let $\mathsf{NS}[i, B, B', (a, k), t]$ be *true* if there is an assignment $\pi : N \to A$ where

(i) the set of activities assigned to players in $\text{desc}(i)$ is exactly B';

(ii) player i is assigned to a and is in a coalition with k other players;

(iii) exactly t players in $\text{desc}(i)$ belong to the same group as i, so $|\text{desc}(i) \cap \pi_i| = t$;

(iv) the t players in $\text{desc}(i) \cap \pi_i$ weakly prefer (a, k) to $(b, 1)$ for each $b \in A \setminus B$, and have no incentive to deviate to the other groups, i.e., every player in $\text{desc}(i) \cap \pi_i$ whose children do not belong to π_i likes (a, k) at least as much as each of the alternatives she can deviate to;

(v) the players in $\text{desc}(i) \setminus \pi_i$ weakly prefer their alternative under π to engaging alone in any of the activities in $A \setminus B$, have no NS-deviation to activities in $B' \setminus \{a\}$, and have no incentive to deviate to i's coalition, i.e., if $a \neq a_\emptyset$, then every player $j \in \text{desc}(i) \setminus \pi_i$ whose parent belongs to π_i likes $(\pi(j), |\pi_j|)$ at least as much as $(a, k+1)$.

Otherwise, we let $\mathsf{NS}[i, B, B', (a,k), t]$ be *false*. By construction, our instance admits a Nash stable assignment if and only if $\mathsf{NS}[r, B, B', (a,k), k]$ is *true* for some combination of the arguments B, B', and (a,k), where r is the root, because by (iv) and (v) there are then no NS-deviations. Notice that the size of the dynamic programming table is at most $p4^p n^3$. See Figure 3.4a for the high-level idea of the algorithm.

To complete the proof, we show that values $\mathsf{NS}[i, B, B', (a,k), t]$ can be efficiently computed in a bottom-up manner. If i is a leaf, we set $\mathsf{NS}[i, B, B', (a,k), t]$ to *true* if $B' = \{a\}$, $t = 1$, and i weakly prefers (a,k) to every alternative $(b,1)$ such that $b \in A \setminus B$; otherwise, we set $\mathsf{NS}[i, B, B', (a,k), t]$ to *false*.

Now, consider the case when i is an internal vertex. We order the children of i and let $\text{ch}(i) = \{j_1, j_2, \ldots, j_{|\text{ch}(i)|}\}$. Observe that in each feasible assignment $\pi : \text{desc}(i) \to B'$ the coalition of players that engage in the same activity $b \in B' \setminus \{a\}$ is fully contained in a subtree of some child of i. This induces a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$ of the activity set $B' \setminus \{a\}$ and a permutation/indexing of these activity sets so that the activities in P_i are assigned to the subtree of j_i. We will thus go through all possible partitions of $B' \setminus \{a\}$ and their permutations, and try to find a feasible assignment compatible with it. Figure 3.4b illustrates how each subtree is matched to a different activity set.

We first check whether i strictly prefers some alternative $(b,1)$ such that $b \in A \setminus B$ to (a,k); if so, we set $\mathsf{NS}[i, B, B', (a,k), t]$ to *false*. Otherwise, we proceed and guess a partition and indexing $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$ of $B' \setminus \{a\}$. Now we will show that we can determine in polynomial time whether there exists of a stable assignment compatible with this indexed partition.

Claim 3.1. *There exists an algorithm that given a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$ of $B' \setminus \{a\}$, checks whether there exists a feasible assignment $\pi : N \to A$ such that the conditions* (i) *to* (v) *hold, and each activity set in $\mathcal{P}$ is assigned to the players in* $\text{desc}(j)$ *for some $j \in \text{ch}(i)$ in such a way that for every $q \in [|\mathcal{P}|]$ and $c \in [|\text{ch}(i)|]$, P_q is assigned to* $\text{desc}(j_c)$ *only if the prefix $P_1, P_2, \ldots, P_{q-1}$ are assigned to the subtrees* $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$*, finds one if it exists, and runs in time $O(p^2 n^3)$ with respect to the number of activities p.*

Proof. We give a dynamic programming algorithm. Observe that a stable assignment can allocate only activity a or nothing to players before allocating the activity set P_1 to some subtree. For convention, we thus denote P_0 by the empty activity set, which may be assigned

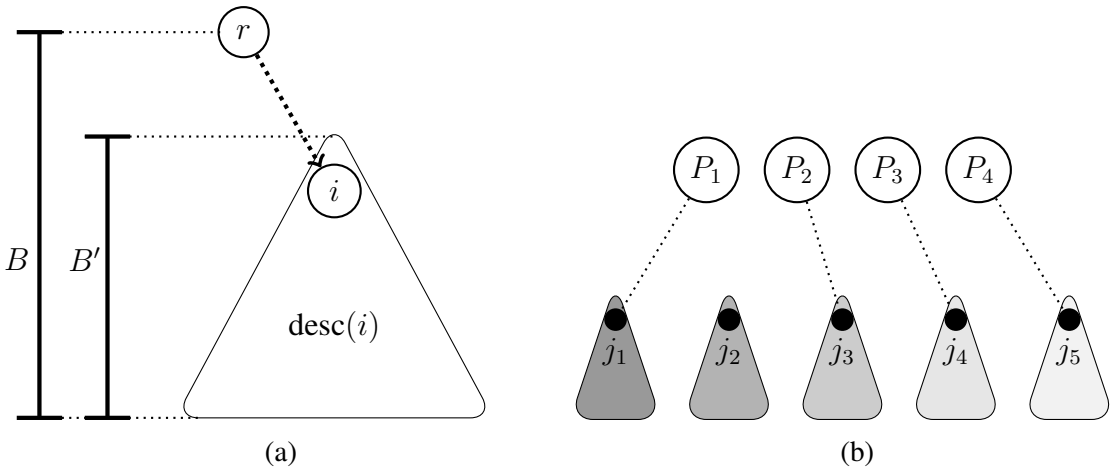


Figure 3.4: Figure 3.4a describes the high-level idea of the algorithm: at each subtree, the algorithm checks whether the descendants desc(i) can be assigned to the activity set B', when the whole tree only uses the activity set B. Figure 3.4b illustrates how each subtree is matched to a different activity set.

to the first c subtrees where the subtree desc(j_{c+1}) is assigned to the activity set P_1. We will now determine for each $c \in [|\mathsf{ch}(i)|]$ and $q \in [0, |\mathcal{P}|]$ whether the activity sets $P_0, P_1, \ldots, P_q$ can be assigned to the subtrees rooted at $j_1, j_2, \ldots, j_c$, and exactly ℓ players can be assigned to the activity a; we refer to this subproblem by $T[j_c, P_q, \ell]$.

We initialize $T[j_1, P_q, \ell]$ to *true* if

- the empty activity set P_0 can be allocated to the first subtree desc(j_1), i.e., $q = 0$, $\ell = 0$, and $\mathsf{NS}[j_1, B, \emptyset, (a_\emptyset, 1), 1]$ is *true*, and ($a = a_\emptyset$ or j weakly prefers $(a_\emptyset, 1)$ to $(a, k+1)$); or
- only the activity a can be assigned to ℓ players in desc(j_1), i.e., $q = 0$, $\ell \geq 1$, and $\mathsf{NS}[j_1, B, \{a\}, (a, k), \ell]$ is *true*;
- only the activity set P_1 can be assigned to players in desc(j_1), i.e., $q = 1$, $\ell = 0$, and there exists an alternative $(b, x) \in P_1 \times [n] \cup \{(a_\emptyset, 1)\}$ such that $\mathsf{NS}[j_1, B, P_1, (b, x), x]$ is *true*, $b = a_\emptyset$ or i weakly prefers (a, k) to $(b, x+1)$, and ($a = a_\emptyset$ or j_1 weakly prefers (b, x) to $(a, k+1)$); or
- P_1 can be assigned to players in desc(j_1), and activity a can be assigned to ℓ players from desc(j_1), i.e., $q = 1$, $\ell \geq 1$, and $\mathsf{NS}[j_1, B, P_1 \cup \{a\}, (a, k), \ell]$ is *true*.

We set $T[j_1, P_q, \ell]$ to *false* otherwise.

Then, we iterate through $j_1, j_2, \ldots, j_{|\mathsf{ch}(i)|}$ and $P_0, P_1, \ldots, P_{|\mathcal{P}|}$, and update $T[j_c, P_q, \ell]$: for each $c \in [|\mathsf{ch}(i)|]$, for each $q \in [0, |\mathcal{P}|]$, and for each $\ell \in [0, t]$, we set $T[j_c, P_q, \ell]$ to *true* if one of the following statements holds:

- $P_1, P_2, \ldots, P_q$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ with ℓ players from the subtrees being assigned to the activity a, and the void activity can be assigned to the subtree $\text{desc}(j_c)$, i.e., both $T[j_{c-1}, P_q, \ell]$ and $\mathsf{NS}[j_c, B, \emptyset, (a_\emptyset, 1), 1]$ are *true*, and $a = a_\emptyset$ or player j_c weakly prefers $(a_\emptyset, 1)$ to $(a, k+1)$;

- $P_1, P_2, \ldots, P_q$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ while the activity a is assigned to x players from the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ and to $\ell - x$ players from the subtree $\text{desc}(j_c)$, i.e., $\ell \geq 2$, and there exists an $x \in [\ell - 1]$ such that $T[j_{c-1}, P_q, \ell - x]$ and $\mathsf{NS}[j_c, B, \{a\}, (a, k), x]$ are *true*;

- $P_1, P_2, \ldots, P_{q-1}$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ with ℓ players from the subtrees being assigned to the activity a, and only the activity set P_q can be assigned to the subtree $\text{desc}(j_c)$, i.e., $T[j_{c-1}, P_{q-1}, \ell]$ is *true*, and there exists an alternative $(b, x) \in P_q \times [n] \cup \{(a_\emptyset, 1)\}$ such that $\mathsf{NS}[j_c, B, P_q, (b, x), x]$ is *true*, ($b = a_\emptyset$ or i weakly prefers (a, k) to $(b, x+1)$), and ($a = a_\emptyset$ or j_c weakly prefers (b, x) to $(a, k+1)$);

- $P_1, P_2, \ldots, P_{q-1}$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ and P_q can be assigned to the subtree $\text{desc}(j_c)$ while the activity a is assigned to x players from the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ and to $\ell - x$ players from the subtree $\text{ch}(j_c)$, i.e., $\ell \geq 2$, and there exists an $x \in [\ell - 1]$ such that both $T[j_{c-1}, P_{q-1}, \ell - x]$ and $\mathsf{NS}[j_c, B, P_q \cup \{a\}, (a, k), x]$ are *true*.

We set $T[j_c, P_q, \ell]$ to *false* otherwise.

Clearly, a desired assignment exists if and only if $T[j_{|\mathsf{ch}(i)|}, P_{|\mathcal{P}|}, t-1]$ is *true*. The size of the dynamic programming table is at most pn^2 and each entry can be filled in $O(pn)$ time. This prove the claim. □

The size of the dynamic programming table $\mathsf{NS}[i, B, B', (a, k), t]$ is at most $4^p pn^3$. For each entry, we consider at most $(p+1)^p$ partitions of $B' \setminus \{a\}$, and for each partition at most $p!$ permutations; further, for each permutation we have shown that there exists an $O(p^2 n^3)$ time algorithm that checks the existence of a stable assignment compatible with it. Hence we conclude that our problem for trees can be solved in $O(4^p (p+1)^p p! p^3 n^6)$ time.

Now, if (N, L) is a forest, we can combine the FPT algorithm described above with the algorithm for graphs with small connected components in the proof of Theorem 3.8. The running time of the latter algorithm is a product of the time required to find a Nash stable assignment for a single connected component and the time required to combine solutions for different components; As we have seen, the former is $O(p^{c+1} c^2)$, where c is the maximum

component size and the latter is $O(2^p n)$. In our case, each connected component is a tree, so instead of the $O(p^{c+1}c^2)$ algorithm for general graphs we can use our FPT algorithm for trees. This shows that our problem is in FPT for arbitrary forests. □

The proof for individual stability is similar and can be found in Appendix A.2.

Theorem 3.11. *The problem of deciding whether an instance of* gGASP *with* $|A^*| = p$ *whose underlying social network* (N, L) *is acyclic has an individually stable feasible assignment is in* FPT *with respect to* p*.*

Core stability turns out to be more computationally challenging than Nash stability and individual stability when the number of activities is small and the graph is a star: we will now show that core stable assignments are hard to find even if there are only two activities and the underlying graph is a star (and thus one cannot expect an FPT result with respect to the number of activities for this setting). Later, we will see that this hardness result can be extended to the case where (N, L) is a clique, i.e., to standard GASP, thereby solving a problem left open in the work of Darmann [41].

Theorem 3.12. *It is* NP*-complete to determine whether an instance of* gGASP *admits a core stable assignment even when the underlying graph is a star and the number of non-void activities is* 2*.*

Proof. Our problem is in NP by Proposition 3.1. To establish NP-hardness, we reduce from the NP-complete problem EXACT COVER BY 3-SETS (X3C) [57]. Given a finite set $V = \{v_1, v_2, \ldots, v_{3k}\}$ and a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of 3-element subsets of V, X3C asks whether $\mathcal{S}$ admits an *exact cover* $\mathcal{S}' \subseteq \mathcal{S}$, i.e., $|\mathcal{S}'| = k$ and $V = \bigcup_{S_i \in \mathcal{S}'} S_i$.

Given an instance of X3C, we construct an instance of gGASP as follows. We define the set of activities to be $A^* = \{a, b\}$. We introduce a center player c, two players x_1 and x_2, and a player S_j for each $S_j \in \mathcal{S}$.

Idea. We will construct the dummies of each $v_i \in V$ and the preferences of players as follows:

1. The players c, x_1, and x_2 form an empty core instance; thus, in order to stabilize them, the center c needs to be assigned to activity a, together with k players from $\mathcal{S}$, and x_2 must be assigned to activity b.

2. If the dummies of v_i and the players corresponding to the sets including v_i (denoted by $\mathcal{S}(v_i)$) are assigned to the void activity, then these players together with the center c and x_2 will form a blocking coalition of size $\beta(v_i)$ and deviate to activity b.

Construction details. For each $v_i \in V$, we let $\mathcal{S}(v_i) = \{ S_j \mid v_i \in S_j \in \mathcal{S} \}$ and $\beta(v_i) = i + 3k + 1$ and create a set $\mathsf{Dummy}(v_i) = \{d_i^{(1)}, d_i^{(2)}, \ldots, d_i^{(\beta(v_i)-|\mathcal{S}(v_i)|-2)}\}$ of dummy players. We attach each of the dummies to the center c. Intuitively, for each $i \in [3k]$ the number $\beta(v_i)$ is the target coalition size when all players in $\mathsf{Dummy}(v_i)$ are engaged in activity b, together with c, x_2, and the players in $\mathcal{S}(v_i)$. Note that the values $\beta(v_i)$ are at least 3 and distinct over $i \in [3k]$. We then attach to the center, the players x_1, x_2, all the players in $\mathcal{S}$, and all the dummy players. The illustration of the graph is presented in Figure 3.5.

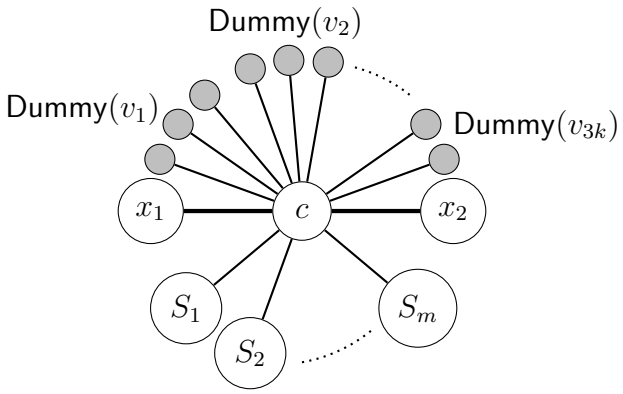


Figure 3.5: The graph constructed in the hardness proof for Theorem 3.12.

Now the players' preferences are defined as follows. For each $S_j \in \mathcal{S}$, we let $B_j = \{b\} \times \{ \beta(v_i) \mid v_i \in S_j \}$; also, set $B = \bigcup_{S_j \in \mathcal{S}} B_j$. The preferences of each player $S_j \in \mathcal{S}$ are given by

$$S_j : (a, k+1) \succ B_j \succ (a_\emptyset, 1).$$

For each $i \in [3k]$ the dummy players in $\mathsf{Dummy}(v_i)$ only approve the alternative $(b, \beta(v_i))$.

Finally, the preferences of x_1, c, and x_2 are given by

$$\begin{aligned} x_1 &: (b,2) \succ (a,3) \succ (a_\emptyset, 1) \\ c &: (a,2) \succ (b,2) \succ (a,3) \succ B \succ (a,k+1) \succ (a_\emptyset, 1) \\ x_2 &: (a,3) \succ B \succ (b,1) \succ (a,2) \succ (a_\emptyset, 1). \end{aligned}$$

Note that the preferences of c, x_1 and x_2, when restricted to $A \times [1,2,3]$, form an instance of gGASP with an empty core (Example 3.2).

Correctness. We will show that $(V, \mathcal{S})$ admits an exact cover if and only if there exists a core stable feasible assignment.

Let $\mathcal{S}'$ be an exact cover in $(V, \mathcal{S})$. Then, we construct a feasible assignment π by assigning activity a to the player c and the players $S_j \in \mathcal{S}'$, assigning b to the player x_2, and

assigning the void activity to the remaining players. Clearly, π is individually rational since $|\mathcal{S}'| = k$ and hence $|\pi^a| = k + 1$. Further, notice that no connected subset Z together with activity a strongly blocks π: by the definition of a blocking coalition, every such subset has to contain the players in $\mathcal{S}'$, who are currently enjoying one of their top alternatives. It remains to show that no connected subset Z together with activity b strongly blocks π. Suppose towards a contradiction that such a subset Z exists; as $x_2 \in H$, it must be the case that $|Z| = \beta(v_i)$ for some $v_i \in V$ and hence Z consists of agents who approve $(b, \beta(v_i))$, i.e., $Z = \{c, x_2\} \cup \mathcal{S}(v_i) \cup \mathsf{Dummy}(v_i)$ for some $v_i \in V$. However, since $\mathcal{S}'$ is an exact cover, there is a set $S_j \in \mathcal{S}' \cap \mathcal{S}(v_i)$ with $\pi(S_j) = a$, and this agent prefers $(a, k + 1)$ to $(b, \beta(v_i))$, a contradiction. Hence, π is core stable.

Conversely, suppose that there exists a core stable feasible assignment π and let $\mathcal{S}' = \{ S_j \in \mathcal{S} \mid \pi(S_j) = a \}$.

We will first argue that $\pi(c) = a$. Indeed, if $\pi(c) = b$ and the only other agent to engage in b is x_1, then $\pi^a = \emptyset$ since no player approves $(a, 1)$; thus, the players c and x_2 can deviate to a. If $\pi(c) = b$ and $|\pi^b| = \beta(v_i)$ for some $v_i \in V$ then $\pi^a = \emptyset$ and c, x_1 and x_2 can deviate to a. If $\pi(c) = a_\emptyset$, then $\pi^a = \emptyset$ and the players c, x_1, and x_2 would deviate to a. It follows that $\pi(c) = a$. Now, if $\pi^a = \{c, x_2\}$ then $\pi^b = \emptyset$ and agent x_2 can deviate to b. Similarly, if $\pi^a = \{c, x_1, x_2\}$ then $\pi^b = \emptyset$, and c and x_1 can deviate to b. If follows that $|\pi^a| = k + 1$ and hence $|\mathcal{S}'| = k$. Also, since there is no coalition assigned to an alternative in B, x_2 must be engaged alone in activity b; otherwise, the player can deviate to b.

Now, if $\mathcal{S}'$ is not an exact cover for $(V, \mathcal{S})$, there exists an element $v_i \in V$ such that no player in $\mathcal{S}(v_i)$ is assigned to a; that is, all players in $\mathcal{S}(v_i)$ are assigned to the void activity. Then the coalition $\{c, x_2\} \cup \mathcal{S}(v_i) \cup \mathsf{Dummy}(v_i)$ together with the activity b strongly blocks π, contradicting the stability of π. Thus, $\mathcal{S}'$ is an exact cover for $(V, \mathcal{S})$. □

The hardness result in Theorem 3.12 immediately generalizes to instances of gGASP with more than two activities: we can modify the construction in our hardness reduction by introducing additional activities that no player wants to engage in. In contrast, checking the existence of core stable assignments in gGASP is easy if $|A^*| = 1$, irrespective of the structure of the social network.

Proposition 3.2. *Every instance of* gGASP *with* $A = \{a, a_\emptyset\}$ *admits a core stable assignment; moreover, such an assignment can be computed in polynomial time.*

Proof. Consider an instance $(N, (\succeq_i)_{i \in N}, A, L)$ of gGASP with $A = \{a, a_\emptyset\}$, and let $n = |N|$. For each $s \in [n]$, let S_s be the set of all players who weakly prefer (a, s) to $(a_\emptyset, 1)$. If for each $s \in [n]$ the largest connected component of S_s with respect to (N, L) contains fewer than s agents, then no outcome in which a non-empty set of agents engages in a is

individually rational, whereas the assignment π given by $\pi(i) = a_\emptyset$ for all $i \in N$ is core stable. Otherwise, consider the largest value of s such that S_s has a connected component of size at least s. Find a connected subset of S_s of size exactly s, and assign the agents in this subset to a; assign the remaining agents to $a_\emptyset$. To see that this assignment is core stable, note that for every deviating coalition S we would have $|S| > s$, and hence such a coalition is either disconnected or some players in S prefer $(a_\emptyset, 1)$ to $(a, |S|)$. □

It is not clear if the problem of finding core stable assignments in gGASP is in FPT with respect to the number of activities when the underlying graph is a path. However, we can place this problem in XP with respect to this parameter. In fact, we have the following more general result for graphs with few connected coalitions (see the work of Elkind [52] for insights on the structure of such graphs).

Proposition 3.3. *Given an instance $(N, (\succeq_i)_{i\in N}, A, L)$ of* gGASP *with $|N| = n$, $|A| = p + 1$, such that the number of non-empty connected subsets of (N, L) is κ, we can decide in time $O(\kappa^p) \cdot \mathrm{poly}(n, p)$ whether this instance admits a core stable assignment, and find one such assignment if it exists.*

Proof. Let $C_1, \ldots, C_\kappa$ be the list of all non-empty connected subsets of (N, L). Since each assignment of players to activities has to assign a connected (possibly empty) subset of players to each activity, we can bound the number of possible assignments by $(\kappa + 1)^p$: each of the p non-void activities is assigned to a coalition in our list or to no player at all, and the remaining players are assigned the void activity. We can then generate all these assignments one by one and check if any of them is core stable; by Proposition 3.1, the stability check can be performed in time polynomial in n and p. □

If the social network (N, L) is a path, it has $O(|N|^2)$ connected subsets. Thus, we obtain the following corollary.

Corollary 3.1. *The problem of deciding whether a given instance of* gGASP *whose underlying graph is a path admits a core stable assignment is in* XP *with respect to the number of activities.*

3.4.3 Cliques

It is unlikely that our FPT results can be extended to even cliques: our next result shows that the problem of finding a Nash stable outcome is W[1]-hard with respect to the number of activities even for classic GASP, i.e., when the social network imposes no constraints on possible coalitions and players have approval preferences.

Theorem 3.13. *The problem of determining whether an instance of* gGASP *admits a Nash stable feasible assignment is* W[1]*-hard with respect to the number of activities, even if the underlying graph (N, L) is a clique and each player has an approval preference.*

Proof. We will provide a parameterized reduction from CLIQUE on regular graphs. Recall that given an undirected graph $G = (V, E)$ and a positive integer k, CLIQUE asks if G admits a clique of size k. The problem is known to be W[1]-hard with respect to k even for *regular graphs*, i.e., graphs where each vertex has the same degree (see e.g. Theorem 13.4 in the book of Cygan et al. [40]).

Given a regular graph $G = (V, E)$ and a positive integer k, where $|V| = n$, $|E| = m$, and each vertex of G has degree $\delta \geq k - 1$, we create an instance of gGASP whose underlying graph is a clique, as follows. We create one *vertex activity* a_i for each $i \in [k]$, one *edge activity* b_j for each $j \in [k(k-1)/2]$, and two other activities c and d. The set of non-void activities is given by

$$A^* = \{\, a_i \mid i \in [k] \,\} \cup \{\, b_j \mid j \in [k(k-1)/2] \,\} \cup \{c, d\}.$$

For each $v \in V$, we create one *vertex player* v, and for each edge $e = \{u, v\} \in E$, we create two *edge players* e_{uv} and e_{vu}.

Idea. We will create dummies of vertices and edges in such a way that a stable assignment has the following properties:

1. if a vertex player v is assigned to a vertex activity, it forms a coalition of size $\alpha(v)$ that consists of the player v, its dummies, and $\delta - k + 1$ edge players incident to the vertex; and

2. if an edge player e_{vu} is assigned to an edge activity, it forms a coalition of size $\beta(e)$ that consists of the edge player e_{uv}, e_{vu}, and the dummies of the edge $\{u, v\}$.

Construction details. Let $P = \{2j + n \mid j \in [n]\}$, and let $\alpha : V \to P$ be a bijection that assigns a distinct number in P to each vertex $v \in V$. Note that $u \neq v$ implies that $\alpha(u)$ and $\alpha(v)$ differ by at least 2, i.e., $\alpha(u) + 1 \neq \alpha(v)$ and $\alpha(v) + 1 \neq \alpha(u)$. Similarly, let $Q = \{2j \mid j \in [m]\}$ and let $\beta : E \to Q$ be a bijection that assigns a distinct number in Q to each edge $e \in E$. For each $v \in V$ we construct a set $\mathsf{Dummy}(v)$ of $\alpha(v) - \delta + k - 2$ *dummy vertex players*. Similarly, for each $e \in E$ we construct a set $\mathsf{Dummy}(e)$ of $\beta(e) - 2$ *dummy edge players*. Lastly, we create one stalker s. The set of players is given by

$$N = V \cup \bigcup_{v \in V} \mathsf{Dummy}(v) \cup E \cup \bigcup_{e \in E} \mathsf{Dummy}(e) \cup \{s\}.$$

Now the preferences are given as follows.

- Each vertex player $v \in V$ approves the alternatives $(c, n-k)$ and $(d, 1)$ as well as each alternative $(a_i, \alpha(v))$ where $i \in [k]$.
- Each dummy in $\mathsf{Dummy}(v)$ of a vertex player v approves the alternatives $(a_i, \alpha(v))$ and $(a_i, \alpha(v)+1)$ where $i \in [k]$.
- Each edge player e_{vu} approves the alternatives $(a_i, \alpha(v))$ and $(a_i, \alpha(v)+1)$ where $i \in [k]$ as well as the alternatives $(b_j, \beta(e))$ where $j \in [k(k-1)/2]$.
- The dummies in $\mathsf{Dummy}(e)$ only approve the alternatives $(b_j, \beta(e))$ where $j \in [m]$.
- The stalker s approves the alternative $(d, 2)$.

All of these players are indifferent among all alternatives they approve. Finally, we take the underlying social network to be a complete graph. Note that the number of activities depends on k, but not on n, and the size of our instance of gGASP is bounded by $O(n^2 + m^2)$.

Correctness. We will now argue that the graph G contains a clique of size k if and only if there exists a Nash stable assignment for our instance of gGASP.

Suppose that G contains a clique S of size k. We construct an assignment π as follows. We assign the activity c to all vertex players who do not belong to S. The size of this coalition is $n-k$. We then assign the vertex activities to the rest of vertex players and its dummies: We take some bijection η between S and $[k]$, and for each $v \in V$ we form a coalition of size $\alpha(v)$ that engages in $a_{\eta(v)}$: this coalition consists of v, all players in $\mathsf{Dummy}(v)$, and all edge players e_{vu} such that $u \notin S$. Lastly, we assign the edge activities to the edge players both of whose endpoints are in S: we establish a bijection ξ between the edge set $\{\, e = \{u, v\} \in E \mid u, v \in S \,\}$ and $[k(k-1)/2]$, and assign the activity $b_{\xi(e)}$ to the edge players e_{uv} and e_{vu} as well as to all players in $\mathsf{Dummy}(e)$. All other players, namely the stalker s as well as the remaining edge players and dummy players, are assigned the void activity. We will now argue that the resulting assignment π is Nash stable.

Clearly, no player assigned to an activity a_i, b_j, or c wishes to deviate. Now, consider a dummy of some vertex v that is assigned to the void activity. By construction, we have $\pi(v) = c$. The dummy only wants to join a coalition if there is a coalition which engages in an activity a_i and whose size is $\alpha(v) - 1$ or $\alpha(v)$; however, no such coalition exists. Similarly, consider an edge player e_{vu} with $\pi(e_{vu}) = a_\emptyset$. We have $v \notin S$, and therefore e_{vu} does not want to join any of the existing coalitions because they have the wrong sizes; the same argument applies to all the dummies of e. Further, the stalker s does not want to deviate since there is no coalition of size 1 that engages in an activity d. Hence, π is Nash stable.

Conversely, suppose that there exists a Nash stable feasible assignment π. By individual rationality of π, every player engages either in the void activity or in an approved alternative. Notice that if a vertex player v were assigned to the activity d, the stalker s would have an NS-deviation to d; thus, no player is assigned to d. If a vertex player v were assigned to the void activity, then v would have an NS-deviation to activity d. Thus, π cannot allocate vertex players to either d or the void activity. This means that π must allocate the activity c to $n-k$ vertex players, and allocate the k vertex activities a_i to the remaining k vertex players v, each together with $\alpha(v)-1$ other players. Now, let $S = \{v \in V \mid \pi(v) = a_i \text{ for some } i \in [k]\}$. By construction, $|S| = k$. We will show that S is a clique. That is, we prove the following claim.

Claim 3.2. *For any $v \in S$, v is adjacent to every other vertex in S, i.e., there are $k-1$ edge players e_{vu} with $u \in S$.*

Proof. Consider a player $v \in S$, and let $\pi(v) = a_i$. Observe that by individual rationality $|\pi^{a_i}| = \alpha(v)$; thus, all players in $\mathsf{Dummy}(v)$ are assigned to a_i since otherwise they could deviate to a_i. The only other players who approve $(a_i, \alpha(v))$ are edge players e_{vu}. Thus, $\delta - k + 1$ such players must be assigned to a_i, and the remaining $k-1$ of these players must be assigned to some activity b_j, since otherwise they would deviate to a_i. It remains to show that the $k-1$ edge players assigned to some b_j are incident to vertices in S. For each $v \in S$, consider the set of players $E_{\pi,v} = \{e_{vu} \mid v \in S, \pi(e_{vu}) = b_j \text{ for some } j \in [k(k-1)/2]\}$. We have argued that $|E_{\pi,v}| = k-1$ for each v. Thus, $|\bigcup_{v\in S} E_{\pi,v}| = k(k-1)$.

Now, let E_π be the set of all edge players who are assigned to edge activities b_j. Clearly $\bigcup_{v\in S} E_{\pi,v} \subseteq E_\pi$. Note that by individual rationality, each edge activity b_j can be assigned to at most two edge players simultaneously (namely, to the two edge players e_{uv}, e_{vu} corresponding to edge e). Since there are $k(k-1)/2$ edge activities, it follows that $|E_\pi| \leqslant k(k-1)$. Therefore, we must have

$$\bigcup_{v\in S} E_{\pi,v} = E_\pi. \tag{3.2}$$

Now, consider an edge player e_{vu} with $v \in S$ and $\pi(e_{vu}) = b_j$ for some $j \in [k(k-1)/2]$. By individual rationality we have $|\pi^{b_j}| = \beta(\{u, v\})$ and hence π^{b_j} consists of e_{vu}, e_{uv} and all dummies of the edge $\{u, v\}$. By (3.2), we have $e_{uv} \in \bigcup_{v\in S} E_{\pi,v} \subseteq E_\pi$, and so we see that $u \in S$. □

Hence, we conclude that S is a clique of size k. □

We note that if players have approval preferences, individually and core stable outcomes can be computed in polynomial time for any network structure: one can repeatedly find activity a and a maximal unanimous coalition S all of whose members approve a, assign S to a, and remove the activity a and the players in S from consideration. However, if we allow full preference orders, deciding the existence of an individually stable outcome is W[1]-hard with respect to the number of activities for standard GASP i.e., the graph is a clique. The proof can be found in Appendix A.2.

Theorem 3.14. *The problem of determining whether an instance of* gGASP *admits an individually stable assignment is* W[1]*-hard with respect to the number of activities, even if the underlying graph* (N, L) *is a clique.*

On the positive side, for gGASPs on cliques, a Nash stable assignment turns out be computable in polynomial time if the number of activities is a constant. However, it is open if this result can be extended to general gGASPs.

Theorem 3.15. *There exist an algorithm that, given an instance* $(N, A, (\succeq_i)_{i\in N}, L)$ *of* gGASP *with* $|N| = n$*,* $|A| = p + 1$ *such that* (N, L) *is a clique, determines whether it admits a Nash stable assignment in time* $(n + 1)^{p+1}O(np(n + p))$.

Proof. For every mapping $f : A \to [0, n]$ where $\sum_{a\in A} f(a) = n$, we will check if there is a Nash stable assignment π such that $|\pi^a| = f(a)$ for each $a \in A^*$ and $|\{\, i \in N \mid \pi(i) = a_\emptyset \,\}| = f(a_\emptyset)$. There are at most $(n + 1)^{p+1}$ such mappings; hence, it remains to show that each check will take at most $\mathrm{poly}(n)$ steps.

Fix a mapping $f : A \to [0, n]$ with $\sum_{a\in A} f(a) = n$. We construct an instance of the network flow problem as follows. We introduce a source s, a sink t, a node i for each player $i \in N$, and a node a for each activity $a \in A$. We create an arc with unit capacity from the source s to each player, and an arc with capacity $f(a)$ from node $a \in A$ to the sink t. Then, for each $i \in N$ we create an arc of unit capacity from player i to an activity $a \in A$ if and only if

- $a \neq a_\emptyset$, and i weakly prefers $(a, f(a))$ to $(a_\emptyset, 1)$ and to all pairs of the form $(b, f(b) + 1)$, where $b \in A^* \setminus \{a\}$; or
- $a = a_\emptyset$, and i weakly prefers $(a_\emptyset, 1)$ to all pairs of the form $(b, f(b) + 1)$, where $b \in A^*$.

It can be easily verified that an integral flow of size n in this network corresponds to a Nash stable assignment where exactly $f(a)$ players are engaged in each activity $a \in A$. It remains to note that one can check in $O(|V||E|)$ time whether a given network (V, E) admits a flow of a given size [79]. □

A similar argument applies to individual stability: finding an individually stable assignment can be done in polynomial time if the number of activities is a constant.

Theorem 3.16. *There exist an algorithm that, given an instance* $(N, A, (\succeq_i)_{i\in N}, L)$ *of* gGASP *with* $|N| = n$, $|A| = p + 1$ *such that* (N, L) *is a clique, determines whether it admits an individually stable assignment in time* $2^p(n+1)^{p+1}O(np(n+p))$.

Proof. For every mapping $f : A \to [0, n]$ and $g : A^* \to \{0, 1\}$, we will check if there is an individually stable assignment such that

(i) for each $a \in A^*$, $|\pi^a| = f(a)$, and $|\{\, i \in N \mid \pi(i) = a_\emptyset \,\}| = f(a_\emptyset)$;

(ii) for each $a \in A^*$ with $g(a) = 1$, π^a contains a player i who does not want to increase the size of a group, that is, who strictly prefers $(a, f(a))$ to $(a, f(a) + 1)$; and

(iii) for each $a \in A^*$ with $g(a) = 0$, no player wants to deviate to π^a.

There are at most $2^p(n+1)^{p+1}$ such combinations of mappings. We will show that each check will take at most $\mathrm{poly}(n)$ steps.

Fix a mapping $f : A^* \to [0, n]$ and $g : A^* \to \{0, 1\}$. Let $A_0 = \{\, a \in A^* \mid g(a) = 0 \,\}$ and $A_1 = \{\, a \in A^* \mid g(a) = 1 \,\}$.

We construct an instance of the network flow problem as follows. We introduce a source s, a sink t, a node i for each player $i \in N$, a node a for each $a \in A_0 \cup \{a_\emptyset\}$, and two nodes a and $x(a)$ for each activity $a \in A_1$. We create an arc with unit capacity from the source s to each player, and an arc with capacity $f(a)$ from node $a \in A_0 \cup \{a_\emptyset\}$ to the sink t. For each $a \in A_1$ with $f(a) \geq 1$, we create an arc with capacity $f(a) - 1$ from node a to the sink t, and an arc with unit capacity from node $x(a)$ to the sink t. Then, for each $i \in N$

- we create an arc of unit capacity from player i to a node $a \in A^*$ if and only if i weakly prefers $(a, f(a))$ to $(a_\emptyset, 1)$ and to all pairs of the form $(b, f(b) + 1)$, where $b \in A_0 \setminus \{a\}$;
- we create an arc of unit capacity from player i to a node $x(a)$ where $a \in A_1$ if and only if i strictly prefers $(a, f(a))$ to $(a, f(a) + 1)$, and i weakly prefers $(a, f(a))$ to $(a_\emptyset, 1)$ and to all pairs of the form $(b, f(b) + 1)$, where $b \in A_0 \setminus \{a\}$;
- finally, we create an arc of unit capacity from player i to the void activity $a_\emptyset$ if and only if i weakly prefers $(a_\emptyset, 1)$ to all pairs of the form $(b, f(b) + 1)$, where $b \in A_0$.

It can be easily verified that an integral flow of size n in this network corresponds to an individually stable assignment satisfying the conditions (i) - (iii). Again, one can check in polynomial time whether a given network admits a flow of a given size [79], and hence our theorem follows. □

For core stability, Theorem 3.12 can be extended from stars to cliques; however, our proof for cliques relies on having at least four non-void activities. It remains an interesting open problem whether core stable outcomes of gGASPs on cliques can be found efficiently if the number of activities does not exceed 3. We conjecture that the answer is 'no', i.e., the problem of computing core stable outcomes remains NP-hard for $|A^*| = 2, 3$. The proof for the theorem below can be found in Appendix A.2.

Theorem 3.17. *It is* NP*-complete to determine whether an instance of* gGASP *admits a core stable assignment even if the underlying graph is a clique and the number of activities is* 4*.*

3.5 Few Players

In the previous sections, the parameter that we focused on was the number of activities p. Although we expect this parameter to be small in many realistic settings, there are also situations where players can choose from a large variety of possible activities. Assume, for instance, that a small number of scientists are gathering at the university of a large city which offers plenty of options for social, cultural, or sports activities. Usually, the number of options is much larger than the number of people attending. It is, thus, natural to ask if stability-related problems for gGASP are tractable when the number of players n is small. In this section, we give a negative answer by showing that even classical GASP remains W[1]-hard with respect to the number of players n.

We first observe that for all stability concepts considered in this chapter, the problem of finding a stable feasible assignment is in XP with respect to n: we can simply guess the activity of each player (there are at most $(p + 1)^n$ possible guesses) and check whether the resulting assignment is feasible and stable.

Observation 3.1. *The problem of deciding whether a given instance of* gGASP *admits a core stable, Nash stable, or individually stable assignment is in* XP *with respect to the number of players* n*.*

The following theorem shows that gGASP is not fixed-parameter tractable with respect to n unless FPT = W[1]. This is somewhat surprising, because an FPT algorithm with

respect to n could afford to iterate through all possible partitions of the players into coalitions. Thus, the computational difficulty must arise from deciding which group engages in which activity, rather than from deciding who belongs to which group. Indeed, in the following hardness reduction, there is a *unique* partition of players into coalitions that can potentially lead to a stable assignment; thus, computational hardness comes solely from assigning known coalitions of players to activities.

Theorem 3.18. *The problem of determining whether an instance of* gGASP *whose underlying graph is a clique admits a core stable assignment is* W[1]*-hard with respect to the number of players* n*.*

Proof. We describe a parameterized reduction from the W[1]-hard MULTICOLORED INDEPENDENT SET problem (see, e.g., Corollary 13.8 in the book of [40]). Given an undirected graph $G = (V, E)$, a positive integer $h \in \mathbb{N}$, and a vertex coloring $\phi\colon V \to [h]$, MULTICOLORED INDEPENDENT SET asks whether G admits an h-colored independent set, that is, a size-h vertex subset $Q \subseteq V$ such that no pair of distinct vertices in Q is adjacent and the vertices in Q have h pairwise distinct colors. Without loss of generality, we assume that there are exactly q vertices of each color for some $q \in \mathbb{N}$, and that there are no edges between vertices of the same color.

Let (G, h, ϕ) be an instance of MULTICOLORED INDEPENDENT SET with $G = (V, E)$. For convenience, we write $V^{(i)} = \{v_1^{(i)}, \ldots, v_q^{(i)}\}$ to denote the set of vertices of color i for every $i \in [h]$. We construct our gGASP instance as follows. We have one *vertex activity* v for each vertex $v \in V$, one *edge activity* e for each edge $e \in E$, and four other activities a, b, c, and d.

Idea. We will have one *color gadget* $\mathsf{Color}(i)$ for each color $i \in [h]$ and one empty core gadget N_g. For most of the possible assignments, the empty core gadget will be unstable, unless the following holds:

1. For each color $i \in [h]$, the players from the color gadget select a vertex of color i (by being assigned together to the corresponding vertex activity).
2. For each pair of colors $\{i, j\}$, the corresponding selected vertices are not adjacent.
3. The stabilizer player g forms a coalition with the players in N_g.

If all three conditions hold, then the assignment encodes an h-colored independent set.

Construction details. The color gadget $\mathsf{Color}(i)$, $i \in [h]$ consists of two players $p_1^{(i)}$ and $p_2^{(i)}$ with the following preferences:

$$\begin{aligned}
p_1^{(i)} &: E(v_1^{(i)}) \times \{5\} \succ (v_1^{(i)}, 2) \succ E(v_2^{(i)}) \times \{5\} \succ (v_2^{(i)}, 2) \succ \cdots \\
&\quad \cdots \succ E(v_q^{(i)}) \times \{5\} \succ (v_q^{(i)}, 2) \succ (a_\emptyset, 1), \\
p_2^{(i)} &: E(v_q^{(i)}) \times \{5\} \succ (v_q^{(i)}, 2) \succ E(v_{q-1}^{(i)}) \times \{5\} \succ (v_{q-1}^{(i)}, 2) \succ \cdots \\
&\quad \cdots \succ E(v_1^{(i)}) \times \{5\} \succ (v_1^{(i)}, 2) \succ (a_\emptyset, 1),
\end{aligned}$$

where $E(v)$ denotes the set of activities corresponding to edges incident to vertex v.

The stabilizer g approves each alternative of the form $(e, 5)$ with $e \in E$ and is indifferent among them; she also approves $(d, 4)$, but likes it less than all other approved alternatives:

$$g : E \times \{5\} \succ (d, 4) \succ (a_\emptyset, 1).$$

Finally, N_g consists of three players g_1, g_2, and g_3 with the following preferences:

$$\begin{aligned}
g_1 &: (d, 4) \succ (b, 2) \succ (a, 3) \succ (a_\emptyset, 1), \\
g_2 &: (d, 4) \succ (a, 2) \succ (b, 2) \succ (a, 3) \succ (a_\emptyset, 1), \\
g_3 &: (d, 4) \succ (a, 3) \succ (b, 1) \succ (a, 2) \succ (a_\emptyset, 1).
\end{aligned}$$

In a core stable assignment, the activity d should be assigned to the players in N_g together with the stabilizer g; otherwise, the assignment would not be stable as we have seen in Example 3.2.

Together, there are $2h + 4$ players, namely

$$N = \bigcup_{i \in [h]} \mathsf{Color}(i) \cup \{g\} \cup N_g.$$

We take the underlying social network to be a complete graph.

Correctness. We will now argue that the graph G admits an h-colored independent set if and only if our instance of gGASP admits a core stable feasible assignment.

Suppose that there exists an h-colored independent set H. We will construct a core stable assignment π as follows: We assign the players of the color gadget $\mathsf{Color}(i)$ to the activity corresponding to the vertex of color i from H, and we assign the activity d to players in N_g and the stabilizer g. This assignment is core stable: Clearly, no player from N_g wishes to deviate since they are assigned to their top alternatives. Consider the players of color gadget $\mathsf{Color}(i)$. They cannot deviate to another vertex activity, since they have completely opposed preferences over vertex activities of size 2. Now suppose towards a contradiction that there is a coalition S that blocks π together with some edge activity $e = \{v_{\ell_i}^{(i)}, v_{\ell_j}^{(j)}\}$.

Activity e is approved by exactly 5 players, and only with size 5. Thus, S must consist of exactly these 5 players, namely the stabilizer g and the players in $\mathsf{Color}(i)$ and $\mathsf{Color}(j)$. Now we see that the edge e must be incident to the vertex activity assigned to the two players in $\mathsf{Color}(i)$ (otherwise one of these players would prefer their current alternative to $(e, 5)$) and similarly e must be adjacent to the vertex activity assigned to the players in $\mathsf{Color}(j)$. However, this means that there is a pair of adjacent vertices in H, contradicting the fact that H is an independent set. We conclude that π is a core stable feasible assignment.

Conversely, suppose that there exists a core stable feasible assignment π. Then, by core stability, the stabilizer g as well as the players in N_g must be assigned to the activity d. Now, consider some edge activity corresponding to some edge $e = \{v_{\ell_i}^{(i)}, v_{\ell_j}^{(j)}\}$. Recall that e is approved by exactly five players: two player from color gadget $\mathsf{Color}(i)$, two players from color gadget $\mathsf{Color}(j)$, and the stabilizer g. However, all five players approve the edge activity only of size five and g does engage in activity d and not in e. Thus, for every $i \in [h]$, the two players from $\mathsf{Color}(i)$ must be assigned to some vertex activity in $V^{(i)}$. Now, let $H = \{\, v \mid \pi(p_1^{(i)}) = v \text{ for some } i \in [h] \,\}$. We have argued that $|H| = h$. It remains to show that H is an independent set. Suppose towards a contradiction that there are vertices $v_{\ell_i}^{(i)}$ and $v_{\ell_j}^{(j)}$ in H that are adjacent. Notice that the players in $\mathsf{Color}(i)$ strictly prefers the edge activity $e = \{v_{\ell_i}^{(i)}, v_{\ell_j}^{(j)}\}$ with size 5 to their vertex activity with size 2. Similarly, the players in $\mathsf{Color}(j)$ as well as the stabilizer g strictly prefer the edge activity e with size 5 to their alternatives. Together with the activity e, these players can block π, a contradiction to the core stability of π. □

The proofs for Nash and individually stability are related to the above reduction, but technically more involved and can be found in Appendix A.2.

Theorem 3.19. *The problem of determining whether an instance of* gGASP *whose underlying graph is a clique admits a Nash stable or individually stable assignment is* W[1]*-hard with respect to the number of players* n*.*

Note that although we showed W[1]-hardness for each of the parameters p and n, parameterizing by the combined parameter $p + n$ immediately gives fixed-parameter tractability, since the input size is trivially upper-bounded by $n^2 \cdot p$.

3.6 Summary and Related Work

We have initiated the study of group activity selection problems with network structure, and found that even for very simple families of graphs, computing stable outcomes is NP-hard. We identified several ways to circumvent this computational intractability. For gGASPs

with copyable activities, we showed that there exists a polynomial time algorithm to compute stable outcomes. We then investigated the parameterized complexity of computing stable outcomes of group activity selection problems on networks, with respect to two natural parameters. Many of our hardness results hold for the standard GASP, where there are no constraints on possible coalitions; however, some of our positive results only hold for acyclic graphs. We summarize our complexity results in Table 3.1.

Interestingly, one of our tractability results holds for GASP, but it is not clear if it can be extended to gGASP; thus, while simple networks may decrease complexity, allowing for arbitrary networks may have the opposite effect. However, counterintuitively, for core stability we obtain hardness with just 2 activities when the underlying network is simple (a star), whereas for cliques our construction uses 4 activities. It is not clear if this is an artefact of our proof approach, or whether finding core stable outcomes is genuinely easier for cliques than for stars.

	general case	few activities (p)	few players (n)	copyable activities
Nash stability and individual stability				
cliques	NP-c.	W[1]-h. (Th. 3.13, 3.14)	W[1]-h. (Th. 3.19)	NP-c. ([15])
acyclic	NP-c. (Th. 3.4, 3.7)	FPT (Th. 3.10,3.11)	XP (Obs. 3.1)	P (Th. 3.3, 3.2)
paths	NP-c. (Th. 3.4, 3.7)	FPT (Th. 3.10,3.11)	XP (Obs. 3.1)	P (Th. 3.3, 3.2)
stars	NP-c. (Th. 3.5, 3.7)	FPT (Th. 3.10,3.11)	XP (Obs. 3.1)	P (Th. 3.3, 3.2)
small comp.	NP-c. (Th. 3.6, 3.7)	FPT (Th. 3.8)	XP	P (Th. 3.3, 3.2)
core stability				
cliques	NP-c.	NP-c. for $p = 4$ (Th. 3.17)	W[1]-h. (Th. 3.18)	NP-c. ([15])
acyclic	NP-c. (Th. 3.7)	NP-c. for $p = 2$ (Th. 3.12)	XP (Obs. 3.1)	P
paths	NP-c. (Th. 3.7)	XP (Prop. 3.3)	XP (Obs. 3.1)	P
stars	NP-c. (Th. 3.7)	NP-c. for $p = 2$ (Th. 3.12)	XP (Obs. 3.1)	P
small comp.	NP-c. (Th. 3.7)	FPT (Th. 3.9)	XP (Obs. 3.1)	P

Table 3.1: Overview of our complexity results. All W[1]-hardness results are accompanied by XP-membership proofs. For all 'XP'-entries, the question whether the problem is fixed-parameter tractable remains open.

Related work Darmann et al. [42] initiate the study of GASP. In the work of Darmann [42], players are assumed to have approval preferences, and a particular focus is placed on individually rational assignments with the maximum number of participants and Nash stable assignments. They obtained a number of complexity results of computing these outcomes while concerning special cases where players have increasing/degreasing preferences on the size of a group. Subsequently, Darmann [41] investigated a model where players submit ranked ballots. In this work, stability concepts such as core stability and individual stability have been adapted from the hedonic games literature. Further, Lee and Shoham [71] studied

stable invitation problems, which is a subclass of GASP where only one activity is assigned to players. This problem was inspired by settings in which an organizer of an event chooses a *stable* set of guests who have preferences over the number of participants of the event. We refer the reader to the recent survey by Darmann [43] for the relation between these models.

Recently, the parameterized aspects of GASP have been considered by several authors. Lee and Williams [72] studied the complexity of standard GASP, with parameter being the number of groups. They showed that computing a maximum individually rational assignment is in FPT with respect to that parameter; however, they proved that this does not extend to other solution concepts, such as Nash stability and envy-freeness, by obtaining a number of W[1]-hardness results. More recently, Gupta et al. [58] investigated the parameterized complexity of finding Nash stable outcomes in the context of gGASP. In their work, computation of a Nash stable outcome was shown to be in FPT with respect to the combined parameters: the maximum size of a group and the maximum degree in the underlying social network. They also presented an FPT algorithm with respect to the number of activities when the underlying network has a bounded tree-width. This generalizes our FPT result for trees and improves the bound on the running time.

GASP are closely related to hedonic games [16, 23]. Much work has been devoted to the complexity study of hedonic games when there is no restriction on coalition formation (see e.g. Woeginger [96] and Aziz and Savani [10]). In particular, the copyable setting of GASP includes a class of anonymous hedonic games where players' preferences are only determined by the size of the coalition to which they belong. Ballester [15] showed that computing Nash, core, and individually stable outcomes of the game is NP-hard for anonymous games; this translates into the NP-hardness of these solutions for copyable instances of gGASP when the social network is a clique. Nevertheless, our positive results for copyable activities imply that in anonymous hedonic games, one can compute a stable outcome in polynomial time if the underlying social network is acyclic.

It is worth mentioning that models with graph connectivity constraints have been studied in different settings from ours [94, 91]. Talmon [94] considered the multiwinner problem when each winner has to represent a connected voting district. In the work of Talmon [94], a similar hardness result concerning optimal committees for paths was obtained; further, computing an optimal committee on graphs with bounded tree-width was shown to be polynomial-time solvable for *non-unique* variants of the problem where several connected districts can be represented by the same winner. This restriction corresponds to our copyable cases of gGASP.

Chapter 4

Learnability in Hedonic Games

So far we have assumed full knowledge of the underlying game. In particular, the algorithm needs to have either *oracle access to players' preferences*, or *structural knowledge of the underlying preference structure* (e.g. some concise representation of players' preferences that one can leverage in order to obtain a poly-time algorithm), in order to construct a stable outcome. Both assumptions are not realistic in practice; eliciting user preferences is often difficult, especially over combinatorially complex domains such as all possible player subsets. How can we find a stable coalition structure when players' preferences are unknown?

A novel relaxation of the core was recently proposed by Balcan et al. [12] in the context of cooperative transferable utility games, and subsequently extended to hedonic games by Sliwinski and Zick [88]. In this framework, one assumes that players' coalitional valuations are drawn from some unknown distribution, which are then used to construct *probably approximately* core stable outcomes; the associated stability concept is referred to as PAC stability. In this chapter, we explore the ability to construct PAC stable outcomes in graph-restricted hedonic games.

In what follows, we will assume that players' preferences are restricted by an underlying network structure: players prefer connected coalitions over disconnected coalitions. Formally, given a graph (N, L) over a finite set of players N, we consider a hedonic game $(N, (\succeq_i)_{i \in N})$ such that each player i prefers $\{i\}$ to any disconnected coalition $S \in \mathcal{N}(i) \setminus \mathcal{F}_L(i)$. Within this framework, we show that if players' preferences are restricted by a forest, one can compute a PAC stable outcome using only a polynomial number of samples. PAC stabilizability holds even if the underlying forest structure is not known in advance, since one can find an approximate forest structure that is likely to be consistent with the true graph structure.

The positive result for trees is interesting in several respects. First, while one can find PAC stable outcomes in polynomial time, computing stable outcomes for hedonic games on

trees is computationally intractable (see Theorem 2.5 in Chapter 2). Second, unlike the work of Sliwinski and Zick [88], we do not require that players' preferences are provided in the form of numerical utilities over coalitions; this not only makes our results more general, but also more faithful to the original hedonic games model, which assumes ordinal information about players' preferences, rather than cardinal utilities.

Further, in Section 4.3.1, we prove a non-trivial technical result on learning tree structures that is of independent interest.

4.1 PAC Learning

We provide a brief introduction to PAC learning. The basic idea is as follows: we are given an unknown function $v : 2^N \to \mathbb{R}$ (a *target concept*) that assigns values to subsets of players; we wish to estimate v on subsets we did not observe. To facilitate this, we assume that v belongs to a *hypothesis class* $\mathcal{H}$ (say, we know that v is an additive valuation). Our goal is to output a *hypothesis* $v^* \in \mathcal{H}$ (e.g. if v is additive, v^* should be as well) that is likely to match the outputs of v on future observations drawn from some distribution $\mathcal{D}$. More formally, a hypothesis v^* is ϵ *approximately correct* w.r.t a probability distribution $\mathcal{D}$ over 2^N and an unknown function v if

$$\Pr_{S\sim\mathcal{D}}[v^*(S) \neq v(S)] < \epsilon.$$

A learning algorithm $\mathcal{A}$ takes as input m samples

$$(S_1, v(S_1)), (S_2, v(S_2)), \ldots, (S_m, v(S_m))$$

drawn i.i.d. from a distribution $\mathcal{D}$ over 2^N, and two parameters $\epsilon, \delta > 0$.

A class of functions $\mathcal{H}$ is (ϵ, δ) *PAC (probably approximately correctly) learnable* if there exists an algorithm $\mathcal{A}$ that for any $v \in \mathcal{H}$ and a probability distribution $\mathcal{D}$ over 2^N, with probability of at least $1 - \delta$, it outputs a hypothesis v^* that is ϵ approximately correct with respect to $\mathcal{D}$ and v. If this holds for any for any $\epsilon, \delta > 0$, $\mathcal{H}$ is said to be *PAC learnable*; moreover, if the running time of $\mathcal{A}$, and the number of samples m are polynomial in $\frac{1}{\epsilon}, \log\frac{1}{\delta}$ and n, $\mathcal{H}$ is said to be *efficiently PAC learnable*.

The value δ is the *confidence parameter*: intuitively, it is the probability that the random samples drawn from $\mathcal{D}$ do not accurately portray the true sample distribution; for example, if $\mathcal{D}$ is the uniform distribution, then it is possible that we draw the same subset in every one of our m samples, but this is unlikely. The value ϵ is called the *error parameter*: it is the likelihood that our hypothesis v^* does not agree with the target concept v. Not all hypothesis classes are efficiently PAC learnable; learnability is inherently related to the complexity of the hypothesis class. The complexity of real-valued functions is commonly measured using

the notion of *pseudo dimension* [2, Chapter 11]. Given a list of sets $S_1, \ldots, S_m \subseteq N$, and corresponding values $r_1, \ldots, r_m \in \mathbb{R}$ we say that a class of functions $\mathcal{H}$ can *pseudo-shatter* $(S_j, r_j)_{j=1}^m$ if for any labeling $\ell_1, \ldots, \ell_m \in \{0, 1\}$, there is some $v \in \mathcal{H}$ such that $v(S_j) \geqslant r_j$ iff $\ell_j = 1$. The *pseudo-dimension* of $\mathcal{H}$, denoted by $P_{dim}(\mathcal{H})$ is

$$\max\{\, m \mid \exists\, (S_j, r_j)_{j=1}^m \text{ that can be shattered by } \mathcal{H} \,\}.$$

The following well-known theorem relates the pseudo-dimension and PAC learnability.

Theorem 4.1 (Anthony and Bartlett [2])**.** *A class of functions $\mathcal{H}$ is (ϵ, δ) PAC learnable using $m = \text{poly}(P_{dim}(\mathcal{H}, \frac{1}{\epsilon}, \log \frac{1}{\delta}))$ samples if there exists an algorithm such that given m samples $(S_1, v(S_1)), (S_2, v(S_2)), \ldots, (S_m, v(S_m))$ drawn i.i.d. from a distribution $\mathcal{D}$, it outputs $v^* \in \mathcal{H}$ consistent with the sample, i.e. $v^*(S_j) = v(S_j)$ for all sampled S_j, and runs in time polynomial in $\frac{1}{\epsilon}, \log \frac{1}{\delta}$ and n. Furthermore, if $P_{dim}(\mathcal{H})$ is superpolynomial in n, $\mathcal{H}$ is not efficiently PAC learnable.*

In other words, in order to establish the PAC learnability of some hypothesis class, it suffices that one shows that its pseudo dimension is low, and that there exists some efficient algorithm that is able to output a hypothesis v^* which matches the outputs of v on all samples. We note that even if an efficient consistent algorithm does not exist (e.g. if the problem of matching a hypothesis to the samples is computationally intractable), a low pseudo dimension is still desirable: it implies that the number of samples needed in order to find a good hypothesis is polynomial.

4.1.1 PAC Stabilizability

Sliwinski and Zick [88] introduce a relaxed notion of core stability for hedonic games, which they term *PAC stability* (this term was first used by Balcan et al. [12] for cooperative transferable utility games). In what follows, we will express players' preferences in terms of cardinal utilities. In other words, player i assigns a value $v_i(S) \in \mathbb{R}$ to every coalition $S \in \mathcal{N}(i)$; we write a hedonic game as $(N, \mathcal{V})$ where $\mathcal{V}$ is a collection of functions $v_i : \mathcal{N}(i) \to \mathbb{R}$ for each $i \in N$.

We say that a partition π is ϵ-PAC stable w.r.t. a probability distribution $\mathcal{D}$ over 2^N if

$$\Pr_{S \sim \mathcal{D}}[S \text{ strongly blocks } \pi] < \epsilon.$$

The inputs to our learning algorithms will be samples

$$(S_1, \vec{v}(S_1)), (S_2, \vec{v}(S_2)), \ldots, (S_m, \vec{v}(S_m)),$$

where $S_1, \ldots, S_m \subseteq N$, and $\vec{v}(S_j)$ is a vector describing players' utilities over S_j; that is, $\vec{v}(S_j) = (v_i(S_j))_{i \in S_j}$.

Given an unknown hedonic game $(N, \mathcal{V})$ belonging to some hypothesis class $\mathcal{H}$, a *PAC stabilizing algorithm* $\mathcal{A}$ takes as input m sets $S_1, \ldots, S_m$ sampled i.i.d. from a distribution $\mathcal{D}$, and players' preferences over the sampled sets; in addition, it receives two parameters $\epsilon, \delta > 0$. The algorithm $\mathcal{A}$ *efficiently PAC stabilizes* $\mathcal{H}$, if for any hedonic game $(N, \mathcal{V}) \in \mathcal{H}$, distribution $\mathcal{D}$ over 2^N, and parameters $\epsilon, \delta > 0$, with probability at least $1 - \delta$, $\mathcal{A}$ outputs an ϵ-PAC stable coalition structure. We again require that the number of samples, m, is bounded by a polynomial in n, $\frac{1}{\epsilon}$ and $\log \frac{1}{\delta}$. Similarly, we say that $\mathcal{H}$ is *efficiently PAC stabilizable* if there is some algorithm $\mathcal{A}$ that efficiently PAC stabilizes $\mathcal{H}$.

4.2 Learning Hedonic Graph Games

In what follows we consider the following hypothesis class.

Definition 4.1. For an undirected graph $G = (N, L)$, let $\mathcal{H}_G$ be the class of all hedonic games $(N, \mathcal{V})$ where for each player $i \in N$, $v(\{i\}) = 0$ and player i strictly prefers its singleton to any disconnected coalition $S \in \mathcal{N}(i) \setminus \mathcal{F}_L$, i.e., $v(S) < 0$ for all $S \in \mathcal{N}(i) \setminus \mathcal{F}_L$.

We first present a baseline negative result: fixing a forest G, the hypothesis class is $\mathcal{H}_G$ is not efficiently PAC learnable. When referring to the PAC learnability of any class of hedonic games, we mean inferring some utility function $v_i^* : 2^N \to \mathbb{R}$ for all $i \in N$ that PAC approximates the true utilities of players in N. This approximation guarantee can be interpreted in both an ordinal and cardinal manner. If we are given player i's ordinal preferences, this simply means that v_i^* is consistent with the ordinal preferences; if we are given player i's cardinal utility function v_i, v_i^* should be a PAC approximation of v_i. As Theorem 4.2 shows, even when we are given additional information about the underlying graph interaction network, players' preferences are not PAC learnable.

Theorem 4.2. *For any graph $G = (N, L)$ with exponentially many connected coalitions, the class $\mathcal{H}_G$ is not efficiently PAC learnable.*

Proof. Recall that $\mathcal{F}_L$ is the set of all feasible coalitions over $G = (N, L)$; by assumption, $|\mathcal{F}_L|$ is exponential.Let $\mathcal{H}_i$ be the set of all possible utility functions $v_i : \mathcal{N}(i) \to \mathbb{R}$ satisfying $v_i(\{i\}) = 0$ and $v(S) < 0$ for all disconnected coalition $S \in \mathcal{N}(i) \setminus \mathcal{F}_E$. The utility player i derives from feasible coalitions in G is unrestricted; in particular, one cannot deduce anything about the utility of some feasible coalition $S \in \mathcal{F}_L$, based on other feasible coalitions' utilities. This immediately implies that the set $\mathcal{F}_L$ can be pseudo-shattered by

$\mathcal{H}_i$. Hence $P_{dim}(\mathcal{H}_i)$ is at least exponential, and by Theorem 4.1, $\mathcal{H}_i$ is not efficiently PAC learnable. □

As an immediate corollary, forest interaction structures do not admit PAC learnable preference structures in general; this is even true if G is a star graph.

Corollary 4.1. *Let G be a star graph over n players; then $\mathcal{H}_G$ is not PAC learnable.*

Proof. For a star with n nodes, any coalition containing the center of the star is feasible, hence it has 2^{n-1} feasible coalitions. By Theorem 4.2, hedonic games on forests are not PAC learnable. □

The reason that hedonic games with forest interaction structures are not PAC learnable is that they may have exponentially many feasible coalitions; this, is also the reason that finding a core stable coalition structure for hedonic games with forest interaction structures is computationally intractable (Theorem 2.5 in Chapter 2). However, we now show how one can still exploit the structural properties of forest graph structures to efficiently compute PAC stable outcomes.

4.3 PAC Stabilizability of Hedonic Graph Games

Having established that hedonic games with a forest interaction structure are not, generally speaking, PAC learnable, we turn our attention to their PAC stabilizability. We divide our analysis into two parts. We begin by assuming that the underlying interaction graph structure is known to us; in Section 4.3.1, we show how one can forgo this assumption.

Theorem 4.3. *If $G = (N, L)$ is a forest, $\mathcal{H}_G$ is efficiently PAC stabilizable.*

Proof. We claim that Algorithm 5 PAC stabilizes $\mathcal{H}_G$. It is a modified algorithm introduced in Demange [46] used to find core stable outcomes for forest-restricted hedonic games in the full information setting (see also Algorithm 4 in Chapter 2). Intuitively, instead of identifying the *guaranteed coalition* for each player precisely, Algorithm 5 approximates it. Again, if the input graph (N, L) is a forest, we can process each of its connected components separately, so we can assume that (N, L) is a tree.

The algorithm first transforms (N, L) into a rooted tree with root r by orienting the edges in L towards the leaves. For every player i starting from the bottom to the top, the algorithm identifies $\mathrm{G}(i)$, the best coalition observed in the samples that is entirely contained in i's subtree, such that others in $\mathrm{G}(i)$ prefer to their own best guaranteed coalition; in other words, $\mathrm{G}(i) \succeq_j \mathrm{G}(j)$ for all $j \in \mathrm{G}(i) \setminus \{i\}$. Having identified $\mathrm{G}(i)$ for every $i \in N$, players

are partitioned according to the $\mathrm{G}(i)$'s from top-down. The main concern is to ensure that $\mathrm{G}(i)$ is a good approximation of its full-information counterpart; this is guaranteed by taking a sufficiently large sample $m = \lceil \frac{n}{\epsilon} \log \frac{n}{\delta} \rceil$.

Given player i, let $\mathcal{G}_i$ the collection of coalitions $\mathrm{G}(j)$ for every descendant $j \neq i$ of i, i.e., $\mathcal{G}_i = \{\, \mathrm{G}(j) \mid j \in \mathrm{desc}(i) \setminus \{i\} \,\}$. For each $i \in N$ and each coalition $X \subseteq N$, we let $P_{\mathcal{G}_i}(X)$ mean that $i \in X \subseteq \mathrm{desc}(i)$, X is connected, and every other player j in $X \setminus \{i\}$ weakly prefers X to $\mathrm{G}(j)$.

Algorithm 5: Finding PAC stable outcomes

input : $\epsilon, \delta > 0$, a tree (N, L), a root $r \in N$, a set $\mathcal{S}$ of $m = \lceil \frac{n}{\epsilon} \log \frac{n}{\delta} \rceil$ samples from $\mathcal{D}$, and $\vec{v}(S)$ for $S \in \mathcal{S}$

output : $\pi^{(r)}$

make a rooted tree with root r by orienting all the edges in L;
initialize $\mathrm{G}(i) \leftarrow \emptyset$ and $\pi^{(i)} \leftarrow \emptyset$ for each $i \in N$;
foreach $t = 0, \ldots, \mathrm{height}(r)$ **do**
    foreach $i \in N$ *with* $\mathrm{height}(i) = t$ **do**
        set
        $\mathcal{S}^* = \{\, S \in \mathcal{S} \cap \mathcal{F}_L(i) \mid S \subseteq \mathrm{desc}(i) \wedge v_j(S) \geqslant v_j(\mathrm{G}(j)), \text{ for all } j \in S \setminus \{i\} \,\}$;
        choose $\mathrm{G}(i) \in \mathrm{argmax}\{\, v_i(S) \mid S \in \mathcal{S}^* \cup \{\{i\}\} \,\}$;
        set $\pi^{(i)} \leftarrow \{\mathrm{G}(i)\} \cup \{\, \pi^{(j)} \mid j \in \mathrm{ch}(\mathrm{G}(i)) \,\}$.

Given $\mathcal{G}_i$ and a distribution $\mathcal{D}$, we say that a coalition X is *top-*$\frac{\epsilon}{n}$ for player i, if

$$\Pr_{S \sim \mathcal{D}}[P_{\mathcal{G}_i}(S) \wedge (v_i(S) > v_i(X) \vee \neg P_{\mathcal{G}_i}(X))] \leqslant \frac{\epsilon}{n}.$$

Trivially, for every $\mathcal{G}_i$ the probability of sampling a top-$\frac{\epsilon}{n}$ coalition for player i from $\mathcal{D}$ is at least $\frac{\epsilon}{n}$; moreover, if $\Pr_{S \sim \mathcal{D}}[P_{\mathcal{G}_i}(S)] \leqslant \frac{\epsilon}{n}$, then any coalition is top-$\frac{\epsilon}{n}$.

Intuitively, $\mathrm{G}(i)$ approximates the best coalition i can form with members of the subtree rooted at i. Algorithm 5's objective is to ensure that sampling a coalition S from $\mathcal{D}$ such that $P_{\mathcal{G}_i}(S) \wedge S \succ_i \mathrm{G}(i)$ is unlikely, namely, the probability of seeing S from $\mathcal{D}$ such that S is better for the highest node i in S than $\mathrm{G}(i)$, and every other player in S prefers it to their $\mathrm{G}(j)$, is smaller than ϵ; this is done by examining enough coalitions so as to see some top-$\frac{\epsilon}{n}$ coalition for every player.

Examine what happens if $\mathcal{G}_i$ containing $\mathrm{G}(j)$'s for i's descendants is fixed upfront, i.e. not dependent on the sample. Let us bound the probability that for i, none of the coalitions

in $\mathcal{S}$ are top-$\frac{\epsilon}{n}$:

$$\begin{aligned}\left(1-\frac{\epsilon}{n}\right)^{m} &= \left(1-\frac{\epsilon}{n}\right)^{\lceil \frac{n}{\epsilon}\log\frac{n}{\delta}\rceil} \\ &\leqslant \left(\left(1-\frac{\epsilon}{n}\right)^{\frac{n}{\epsilon}}\right)^{\log\frac{n}{\delta}} < \left(\frac{1}{e}\right)^{\log\frac{n}{\delta}} < \frac{\delta}{n}\end{aligned} \tag{4.1}$$

Note that Inequality (4.1) is true irrespective of what $\mathcal{G}_i$ is. Taking a union bound, the probability that there is some player i such that there is no top-$\frac{\epsilon}{n}$ coalition for i in $\mathcal{S}$ is at most δ. Note that $\mathcal{S}^*$ can end up not containing any coalition (line 5). But then with high confidence, as a special case of the above consideration, every coalition is top-$\frac{\epsilon}{n}$, and the algorithm can pick $\{i\}$.

Now consider an actual run of the algorithm, where the sample $\mathcal{S}$ is drawn, and for every descendant j of i, $\mathrm{G}(j)$ is computed based on $\mathcal{S}$, and then $\mathrm{G}(i)$ is computed based on the same sample. This runs the risk of some underlying dependence between the computation of $\mathrm{G}(i)$ and the $\mathrm{G}(j)$'s that is not accounted for in Inequality (4.1). This can be easily solved by taking a larger number of samples: if we take $m = \lceil \frac{n^2}{\epsilon}\log\frac{1}{\delta}\rceil$ samples, we can just use $\frac{n}{\epsilon}\log\frac{1}{\delta}$ samples to compute each $\mathrm{G}(i)$ and maintain complete independence in the samples. However, the smaller sample size used in Algorithm 5 can still provide us with the same guarantees.

Consider an equivalent reordering of the process: first, for every $i \in N$, determine the number k of connected coalitions S in the sample such that i will be the highest node in S. Then, draw the other $m-k$ coalitions and compute $\mathrm{G}(j)$'s for every descendant j of i; finally, based on this, determine the family $\mathcal{G}_i$. Note that regardless of what $\mathcal{G}_i$ is, each of the undetermined, independently drawn k coalitions have probability of at least $\frac{\epsilon}{n}$ to be top-$\frac{\epsilon}{n}$ for i. Hence, the inequality 4.1 holds even if $\mathcal{G}_i$ and $\mathrm{G}(i)$ are computed based on the same sample of coalitions $\mathcal{S}$.

We are now ready to prove that the coalition structure outputted by Algorithm 5 returns a PAC stable outcome $\pi^{(r)}$. We observe that any coalition included in the returned $\pi^{(r)}$ is a $\mathrm{G}(i)$ for some i. Note that for every $j \in \mathrm{G}(i)$, we have that $v_j(\mathrm{G}(i)) \geqslant v_j(\mathrm{G}(j))$ (line 5). Now, consider any coalition X that strongly blocks $\pi^{(r)}$; let $i = \mathrm{argmax}_{j\in X}\mathrm{height}(j)$. Since X strongly blocks $\pi^{(r)}$,

$$v_j(X) > v_j(\pi^{(r)}(j)) \geqslant v_j(\mathrm{G}(j))$$

for all players $j \in X$. descIn particular, $v_i(X) > v_i(\mathrm{G}(i))$. By construction of $\mathrm{G}(i)$ and Inequality (4.1), $\mathrm{G}(i)$ is top-$\frac{\epsilon}{n}$ for i; that is,

$$\frac{\epsilon}{n} > \Pr_{S\sim\mathcal{D}}[v_i(S) > v_i(\mathrm{G}(i))] \geqslant \Pr_{S\sim\mathcal{D}}[S = X];$$

thus the probability of drawing a coalition such as X from $\mathcal{D}$ is less than $\frac{\epsilon}{n}$. Taking a union bound over all players,

$$\Pr_{X \sim \mathcal{D}}[X \text{ strongly blocks } \pi^{(r)}] < \epsilon;$$

this guarantee holds with confidence $1 - \delta$. □

We conjecture that a similar argument can imply a stronger statement. That is, we can replace 'strongly block' in the definition of PAC stabilizability with 'weakly block' and still obtain PAC stabilizability on trees. We note that in the full information setting, a strict core outcome does not necessarily exist on trees (see Section 2.2.3 in Chapter 2).

Note that step 6 of the Algorithm 5 is the only step that refers to the numerical representation of agent preferences v_i. The algorithm chooses a coalition with maximal utility value v_i out of some set of possible coalitions; in particular, the only thing required for the successful implementation of Algorithm 5 is players' ranking of coalitions in the sample. In other words, the particular numerical representation of players' preferences plays no role. This is a significant difference from the algorithms devised by Sliwinski and Zick [88], where the type of utility representation functions used was crucial for PAC stability.

4.3.1 Inferring the Underlying Forest Interaction Network

Until now, we assume that the underlying interaction network was given to us as input; this is, naturally, an assumption that we would like to forgo. Instead, we now show that if the underlying graph is a forest $T = (N, L)$, it is possible to infer a forest $T^* = (N, L^*)$ that agrees with the original graph with high probability. Let $\mathcal{T}_n$ be the set of all possible forests over n vertices; this is our hypothesis class for guessing an approximate forest.

By Caylay's formula:

$$|\mathcal{T}_n| = n^{n-2} \tag{4.2}$$

We observe the following variant of Theorem 4.1 for finite hypothesis classes.

Theorem 4.4 (Anthony and Bartlett [2])**.** *Let $\mathcal{C}$ be a finite hypothesis class where $\log |\mathcal{C}|$ is polynomial in n. If there exists a polynomial time algorithm that for any $v \in \mathcal{C}$, and samples*

$$(S_1, v(S_1)) \ldots, (S_m, v(S_m))$$

finds a function $v^* \in \mathcal{C}$ consistent with the samples, i.e., $v^*(S_j) = v(S_j)$ for each $j = 1, 2, \ldots, m$, then $\mathcal{C}$ is efficiently PAC learnable.

Since $\log|\mathcal{T}_n| < n \log n$, all we need is to establish the existence of an algorithm to efficiently compute a forest consistent with a given sample. More formally, let T be an unknown forest; we are given a set of m subsets of vertices that are known to be connected, can we find a forest that is consistent with the subsets? The answer to this question is affirmative, and appears in Conitzer et al. [38].

Theorem 4.5 (Conitzer et al. [38])**.** *Let $T = (N, L)$ be a forest. Given a list $S_1, \ldots, S_m$ of connected vertices in T, there exists a poly-time algorithm that outputs a tree T^* where every subset S_j is connected.*

In other words, under our original problem formulation — where one only observes subsets of feasible coalitions and players' preferences over them — it is possible to find a tree structure consistent with the samples. This immediately implies the following corollary.

Corollary 4.2. *Let $\mathcal{H}^* = \bigcup\{\, \mathcal{H}_G \mid G \text{ is a forest} \,\}$ be the class of all hedonic games whose interaction graph is a forest; then $\mathcal{H}^*$ is efficiently PAC stabilizable.*

Theorem 4.5, while interesting in its own right, provides us with only a partial understanding of the problem: if all one is given is positive examples, it is possible to find a tree structure that is consistent with all connected coalitions. In what follows, we study a more general learning model for inferring forest interaction structures, where we allow both positive (connected coalitions) and negative (disconnected coalitions) examples. As we show in Theorem 4.6, introducing the possibility of negative examples makes the problem computationally intractable, even when the underlying tree needs to be a path.

4.3.2 Constructing a Consistent Path

We now argue that deciding whether there exists a forest consistent with both positive and negative examples is computationally intractable; in fact, this claim holds even when the desired forest is a path. This result stands in sharp contrast to known computational results in the literature; indeed, there are several efficient algorithms for such restricted networks when only connected coalitions are taken into account [1] [24, 66, 39, 54, 59, 67, 60].

Specifically, we are given m samples of node subsets $S_1, \ldots, S_m$; each subset S_j is labeled by a function ℓ_G such that

$$\ell_G(S_j) = \begin{cases} 1 & \text{if } S_j \text{ is connected in } G \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

[1] The problem of deciding the existence of a path consistent with connected coalitions is equivalent to the problem of determining whether the intersection graph of a hypergraph is an interval, which is also closely related to testing the consecutive ones property of a matrix (see, e.g. the survey by Dom [50] for more details).

Our objective is to find a tree T^* such that $\ell_{T^*}(S_j) = \ell_G(S_j)$ for all j. The following theorem states that even for paths, it is NP-hard to determine whether a given sample admits such a graph.

Theorem 4.6. *Given a family of subsets $\mathcal{S} \subseteq 2^N$ such that each set in $\mathcal{S}$ is of size at most 3, and a mapping $\ell : \mathcal{S} \to \{1, 0\}$, it is NP-hard to decide whether there exists a path $T^* = (N, L)$ such that $\ell_{T^*}(S) = \ell(S)$ for each $S \in \mathcal{S}$.*

Proof. We will show the hardness by reduction from a restricted version of 3SAT. Specifically, we consider (3,B2)-SAT. Recall that in this version of 3SAT, each clause contains at most 3 literals, and each variable occurs exactly twice positively and twice negatively; this problem is known to be NP-complete [19].
Idea: consider a formula ϕ with a variable set $X = \{x_1, x_2, \ldots, x_n\}$ and clause set $C = \{c_1, c_2, \ldots, c_m\}$, where for each variable $x_i \in X$ we write $x_i(1)$ and $x_i(2)$ for the two positive occurrences of x_i, and $\bar{x}_i(1)$ and $\bar{x}_i(2)$ for the two negative occurrences of x. We will have one *clause gadget* $C_j = \{c_j(1), c_j(2)\}$ for each clause $c_j \in C$ and one variable gadget $V_i = \{v_i(1), v_i(2)\}$ for each variable $x_i \in X$. Most player arrangements will be inconsistent with the pair $(\mathcal{S}, \ell)$ unless the following holds:

- For each clause $c_j \in C$, a literal player contained in a clause c_j connects the players from a clause gadget.
- For each variable $x_i \in X$, either the pair of positive literal players $x_i(1), x_i(2)$ or the pair of negative literal players $\bar{x}_i(1), \bar{x}_i(2)$ connects the players from a variable gadget.

Hence, one can think of the variable gadgets as forcing a path to make a choice between setting x_i *true* and setting x_i *false*; each clause gadget ensures that the resulting assignment is satisfiable.
Construction details: For each variable $x_i \in X$, we introduce two *variable players* $v_i(1)$ and $v_i(2)$, and four literal players

$$x_i(1), x_i(2), \bar{x}_i(1), \bar{x}_i(2),$$

which correspond to the four occurrences of x. For each clause $c_j \in C$, we introduce two *clause players* $c_j(1)$ and $c_j(2)$. Let $k := (4n - m - 2n) + 1 = 2n - m + 1$. We introduce k *garbage collectors* $g_1, g_2, \ldots, g_k$, and two *leaf* players s and t. Intuitively, garbage collectors will be used to connect the literal players that do not appear in any clause or variable gadget.

Our set $\mathcal{S}$ of samples consists of three subfamilies $\mathcal{C}$, $\mathcal{U}_2$, and $\mathcal{U}_3$: the sets in $\mathcal{C}$ correspond to the connectivity constraints, the sets in $\mathcal{U}_2$ correspond to disconnected coalitions of size 2, and the sets in $\mathcal{U}_3$ correspond to disconnected coalitions of size 3.

First, we construct the set $\mathcal{C}$ that constitutes of

- the four pairs $\{s, c_1(1)\}$, $\{c_m(2), v_1(1)\}$, $\{v_n(2), g_1\}$, $\{g_k, t\}$;
- the consecutive pairs $\{c_j(2), c_{j+1}(1)\}$ for $j \in [m-1]$; and
- the consecutive pairs $\{v_i(2), v_{i+1}(1)\}$ for $i \in [n-1]$.

We next construct the negative samples $\mathcal{U}_2$ and $\mathcal{U}_3$ as follows. The family $\mathcal{U}_2$ is the set of all player pairs, except for the following:

- the pairs in $\mathcal{C}$.
- the pairs of a variable player and its corresponding literal player, i.e., the pairs of the form $\{v_i(h), x_i(h)\}$ or $\{v_i(h), \bar{x}_i(h)\}$.
- the pairs of a clause player and a literal player contained in it, i.e., the pairs of the form $\{c_j(h), y\}$ where y is a literal player in a clause c_j.
- the pairs of positive literal players or negative literal players of each variable, i.e., the pairs of the form $\{x_i(1), x_i(2)\}$ or $\{\bar{x}_i(1), \bar{x}_i(2)\}$.
- the pairs of a literal player and a garbage collector, i.e., the pairs of the form $\{x_i(h), g_{k'}\}$ or $\{\bar{x}_i(h), g_{k'}\}$.

In a path consistent with the samples, each variable player can share an edge with its literal player; and each clause player can share an edge with a literal player contained in it.

The family $\mathcal{U}_3$ consists of the following player triples:

- triples of the form $\{v_i(1), x_i(1), y\}$ where $y \neq v_{i-1}(2)$ and $y \neq x_i(2)$, and the triples of the form $\{v_i(1), \bar{x}_i(1), y\}$ where $y \neq v_{i-1}(2)$ and $y \neq \bar{x}_i(2)$; and
- the triples of the form $\{x_i(1), x_i(2), y\}$ where $y \neq v_i(1)$ and $y \neq v_i(2)$, and the triples of the form $\{\bar{x}_i(1), \bar{x}_i(2), y\}$ where $y \neq v_i(1)$ and $y \neq v_i(2)$.

Here $v_0(2) = c_m(2)$ and $c_0(2) = s$. The above constraints mean that if a variable player $v_i(1)$ and its positive literal player $x_i(1)$ (respectively, its negative literal player $\bar{x}_i(1)$) are adjacent, then the player $x_i(1)$ can be only adjacent to the other positive literal player $x_i(2)$

(respectively, the other negative literal player $\bar{x}_i(2)$), which can then be only adjacent to the other variable player $v_i(2)$.

Finally, for each $S \in \mathcal{S}$ we set $\ell(S) = 1$ if and only if $S \in \mathcal{C}$. Note that the number of players in the instance is bounded by $O(n+m)$ and the number of sets in $\mathcal{S}$ is bounded by $O(n^2+m^2)$.

Correctness: We will now show that ϕ is satisfiable if and only if there exists a path consistent with $(\mathcal{S}, \ell)$.

$\Longrightarrow$: Suppose that there exists a truth assignment $f : X \to \{$*true*,*false*$\}$ that satisfies ϕ. First, since f is a satisfiable assignment for ϕ, for each clause gadget $C_j = \{c_j(1), c_j(2)\}$, we can select exactly one literal that satisfies a clause c_j; we connect the literal player with each of the clause players $c_j(1)$ and $c_j(2)$ by an edge. We combine all the clause gadgets by constructing an edge $\{c_j(2), c_{j+1}(1)\}$ for each $j \in [m-1]$. Now, we consider an assignment that gives the opposite values to f, and connect each variable gadget using the literals corresponding to this assignment. Specifically, for each variable gadget $V_i = \{v_i(1), v_i(2)\}$, if x_i is set to *false*, we select its positive literal players and construct a path that consists of three edges $\{v_i(1), x_i(1)\}$, $\{x_i(1), x_i(2)\}$, and $\{x_i(2), v_i(2)\}$; similarly, if x_i that is set to *true*, we select its negative literal players and construct a path that consists of three edges $\{v_i(1), \bar{x}_i(1)\}$, $\{\bar{x}_i(1), \bar{x}_i(2)\}$, and $\{\bar{x}_i(2), v_i(2)\}$. We then create an edge $\{v_i(2), v_{i+1}(1)\}$ for each $i \in [n]$, and merge the variable gadgets all together.

Finally, we construct a path over the rest of players, by aligning the garbage collectors $g_1, g_2, \ldots, g_k$ in increasing order of their index, and putting one of the remaining $k-1 (= 4n-m-2n)$ literal players into each consecutive pair of garbage collectors arbitrarily. We then merge all the paths by creating the four edges $\{s, c_1(1)\}$, $\{c_m(2), v_1(1)\}$, $\{v_n(2), g_1\}$, and $\{g_k, t\}$; see Figure 5.2 for an illustration. It is easy to verify that the resulting graph is a path consistent with the samples.

$\Longleftarrow$: Conversely, suppose that there is a path $T^* = (N, L)$ consistent with $(\mathcal{S}, \ell)$, i.e., for each $S \in \mathcal{S}$, S is connected in T^* if and only if $S \in \mathcal{C}$. Since every pair in $\mathcal{C}$ should be connected, the four pairs $\{s, c_1(1)\}$, $\{c_m(2), v_1(1)\}$,$\{v_n(2), g_1\}$, $\{g_k, t\}$ must form an edge in T^*. Similarly, we have $\{c_j(2), c_{j+1}(1)\} \in E$ for each $j \in [m-1]$; also, $\{v_i(2), v_{i+1}(1)\} \in L$ for each $i \in [n-1]$. Observe that both players s and t must be the leaves of the constructed path since these players are only allowed to have one neighbor; thus, every other player has degree 2. Combining these observations, the definition of $\mathcal{U}_3$ ensures that our path specifies a truth assignment for X.

Lemma 4.1. *For each $i \in [n]$, we have either*

- $\{v_i(1), x_i(1)\}, \{x_i(1), x_i(2)\}, \{x_i(2), v_i(2)\} \in L$*; or*

$$s \text{ —} c_1(1)\text{–}x_1(1)\text{–}c_1(2)\text{–}c_2(1)\text{–}x_1(2)\text{–}c_2(2)\text{–}c_3(1)\text{–}\bar{x}_2(1)\text{–}c_3(2)\text{–}c_4(1)\text{–}\bar{x}_2(2)\text{–}c_4(2)\text{-}$$
$$\text{-}v_1(1)\text{–}\bar{x}_1(1)\text{–}\bar{x}_1(2)\text{–}v_1(2)\text{–}v_2(1)\text{–}x_2(1)\text{–}x_2(2)\text{–}v_2(2)\text{–}v_3(1)\text{–}x_3(1)\text{–}x_3(2)\text{–}v_3(2)\text{-}$$
$$\text{—}\, g_1 \text{—}\bar{x}_3(1)\text{—}\, g_2 \text{—}\bar{x}_3(2)\text{—}\, g_3 \text{——}\ t$$

Figure 4.1: Graph constructed for the formula $\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_3)$ in the proof of Theorem 4.6. The formula is satisfied by the mapping f that assigns the opposite value to the literals connected to each variable gadget $V_i = \{v_i(1), v_i(2)\}$, i.e., $f(x_1) =$ *true*, $f(x_2) =$ *false*, and $f(x_3) =$ *false*.

- $\{v_i(1), \bar{x}_i(1)\}, \{\bar{x}_i(1), \bar{x}_i(2)\}, \{\bar{x}_i(2), v_i(2)\} \in L$.

Proof. Take any $i \in [n]$. Since each variable player $v_i(1)$ is adjacent to $v_{i-1}(2)$, the other players who can be adjacent to $v_i(1)$ are its literal players $x_i(1)$ and $\bar{x}_i(1)$ due to the constrains in $\mathcal{U}_2$. First, if players $v_i(1)$ and $x_i(1)$ are adjacent, the player $x_i(1)$ can be only adjacent to $x_i(2)$ since $v_{i-1}(2)$ is already adjacent to $v_i(1)$, which then implies that $x_i(2)$ can be only adjacent to $v_i(2)$ due to the constrains in $\mathcal{U}_3$. Similarly, if $v_i(1)$ and $\bar{x}_i(1)$ are adjacent, $\bar{x}_i(1)$ can be only adjacent to $\bar{x}_i(2)$, which then can be only adjacent to $v_i(2)$. This completes the proof.

□

Now take the truth assignment f that sets the variable x_i to *true* if and only if its negative literal players $\bar{x}_i(1)$ and $\bar{x}_i(2)$ are adjacent to variable players $v_i(1)$ and $v_i(2)$. This assignment can be easily seen to satisfy ϕ. Indeed, for each clause $c_j \in C$, the clause player $c_j(2)$ must be adjacent to a literal player in c_j, since each clause player $c_j(2)$ is adjacent to $c_{j-1}(2)$ and the only other players who can be adjacent to $c_j(2)$ are their literal players contained in it; such a literal player corresponds to an occurrence appearing in the assignment f and satisfies a clause c_j. □

In contrast, it is easy to construct a star consistent with a given sample.

Theorem 4.7. *Given a family of subsets $\mathcal{S} \subseteq 2^N$ and a mapping $\ell : \mathcal{S} \to \{1, 0\}$, one can compute in polynomial time a star $T^* = (N, L)$ such that $\ell_{T^*}(S) = \ell(S)$ for each $S \in \mathcal{S}$, if one exists.*

Proof. Find a player c such that

- c belongs to all the connected coalitions of size at least two, i.e., $c \in S$ for all $S \in \mathcal{S}$ with $\ell(S) = 1$ and $|S| \geqslant 2$; and
- c does not belong to any of the disconnected coalitions, i.e., $c \notin S$ for all $S \in \mathcal{S}$ with $\ell(S) = 0$.

If such a player exists, we connect c and the other players by an edge; the resulting graph is clearly a star consistent with the sample. Otherwise, there is no star consistent with the sample. □

4.4 Summary and Related Work

We have established a strong connection between communication structure and the ability to obtain approximate stability in hedonic games. Essentially, we have shown that if one knows that the underlying communication structure is a forest, then one can obtain PAC stable outcomes. We believe that as a research paradigm, finding an probable stable outcome is a useful and important methodological approach. Indeed, by doing so, we are able to circumvent the stringent assumptions required in order to compute stable outcomes. As a future direction, it would be interesting to see how the results of this chapter extend to other graph-restricted settings such as gGASPs.

Related work Several papers establish learning based game-theoretic solution concepts. Sliwinski and Zick [88] introduce PAC stability in hedonic games, and analyze several common classes of hedonic games. Other works on learning and game theory include learning in cooperative games [12], rankings [14], auctions [11, 13, 75, 76] and noncooperative games [53, 87].

Part III

Fairness in Structured Environments

Chapter 5

Fair Division of a Graph

Imagine that the department of computer science at University X is about to move to a new building. Each research group has preferences over rooms, but it would also be desirable for each group to have a contiguous set of offices, to facilitate communication. This situation can be seen as a problem of fair division (where agents are research groups and items are offices) with an additional connectivity requirement. This constraint could be captured by an undirected graph whose vertices are rooms (items) and there is an edge between two vertices if the respective rooms are adjacent; each agent should obtain a connected piece of this graph.

In this chapter, we study a model for fair division under connectivity constraints where there is a graph capturing the dependency relation between items, and each agent's share has to be connected in this graph, and investigate the complexity of finding good allocations in this framework according to three well-studied solution concepts: proportionality, envy-freeness (in conjunction with completeness), and maximin share guarantee. Besides the example in the first paragraph, our model captures a variety of applications, such as time-sharing a processor where tasks can be switched only at pre-defined times, allocating a set of indivisible land plots, or assigning administrative duties to members of an academic department, where there are dependencies among tasks (e.g., dealing with incoming foreign students has some overlap with preparing study programmes in foreign languages, but not with fire safety).

Recently, several papers have studied the concept of the *maximin share guarantee* (MMS) [31], which captures a desirable property of allocations that is easy to achieve for divisible items via cut-and-choose protocols. For indivisible goods, such allocations need not exist [82, 68]. We prove a strong positive result for our setting: an MMS allocation always exists if the underlying graph is a tree, and can be computed efficiently. Our algorithm is an adaptation of the classic last-diminisher procedure for the divisible case. In contrast, we

provide an example where the underlying graph is a cycle of length 8 and there is no MMS allocation. We believe that these results are useful for developing an intuitive understanding of the concept of MMS; in particular, our example for the cycle is much simpler than known examples of instances with no MMS allocation in the absence of graph constraints.

For proportionality and envy-freeness, we obtain hardness results even for very simple graphs: finding proportional allocations turns out to be NP-hard even for paths, and finding complete envy-free allocations is NP-hard both for paths and for stars. Nevertheless, we also obtain some positive results for these solution concepts. In particular, both proportional and complete envy-free allocations can be found efficiently when the graph is a path and agents can be classified into a small number of *types*, where agents are said to have the same type when they have the same preferences over items. [1] If we assume that not just the number of player types, but the actual number of players is small, we obtain an efficient algorithm for finding proportional allocations on arbitrary trees.

5.1 Model

We study fair allocation of indivisible goods where each allocated bundle is connected in an underlying graph.

Definition 5.1. An instance of the *connected fair division problem (CFD)* is a triple $I = (G, N, \mathcal{U})$ where

- $G = (V, E)$ is an undirected graph,
- $N = \{1, \dots, n\}$ is a set of *players*, or *agents*,
- $\mathcal{U}$ is an n-tuple of utility functions $u_i : V \to \mathbb{R}_{\geqslant 0}$, where $\sum_{v \in V} u_i(v) = 1$ for each $i \in N$.

We refer to elements of V as *items*, and denote the number of items by m.

In what follows, we assume that the number of items m is at least as large as the number of players n, since otherwise at least one player gets nothing. Note that when G is a clique, CFD is equivalent to the classic problem of fair allocation with indivisible items.

For each $X \subseteq V$, we set $u_i(X) = \sum_{v \in X} u_i(v)$, so valuations in this chapter are always additive. Two players $i, j \in N$ are of the *same type* if $u_i(v) = u_j(v)$ for all $v \in V$. We denote the number of player types in a given instance by p.

[1] The same parameter was used by Branzei et al. [30] to obtain results for maximizing social welfare; similar ideas have been used in the context of coalition formation [86, 7].

An *allocation* is a function $\pi : N \to 2^V$ assigning each player a bundle of items. An allocation π is *valid* if for each player $i \in N$ the bundle $\pi(i)$ is connected in G and no item is allocated twice, so that $\pi(i) \cap \pi(j) = \emptyset$ for each pair of distinct players $i, j \in N$. We say that a valid allocation π is

- *proportional* if $u_i(\pi(i)) \geqslant \frac{1}{n}$ for all $i \in N$,
- *envy-free* if $u_i(\pi(i)) \geqslant u_i(\pi(j))$ for all $i, j \in N$,
- *complete* if $\bigcup_{i \in N} \pi(i) = V$, and
- *Pareto-optimal* if there is no *Pareto-improvement* of π, i.e., there is no valid allocation π' such that $u_i(\pi'(i)) \geqslant u_i(\pi(i))$ for all $i \in N$ and $u_j(\pi'(j)) > u_j(\pi(j))$ for some $j \in N$.

Notice that an allocation that gives everybody an empty bundle is envy-free, so, to better express the idea of fairness, the requirement of envy-freeness is typically accompanied by completeness or Pareto-optimality.

We also consider *maximin share (MMS) allocations* [31], adapting the usual definition to our setting as follows. Given an instance $I = (G, N, \mathcal{U})$ of CFD with $G = (V, E)$, let Π_n denote the space of all partitions of V into n connected pieces. The *maximin share guarantee* of a player $i \in N$ is

$$\mathrm{mms}_i(I) = \max_{(P_1,\ldots,P_n) \in \Pi_n} \min_{j \in \{1,\ldots,n\}} u_i(P_j).$$

Note that since we are only taking the maximum over connected partitions, these values may be lower than in the general setting without graph constraints. A valid allocation π is a *maximin share (MMS) allocation* if we have $u_i(\pi(i)) \geqslant \mathrm{mms}_i(I)$ for each player $i \in N$.

We consider the following computational problems that all take an instance $I = (G, N, \mathcal{U})$ of the connected fair division problem as input. For computational purposes, we assume that utility functions take values in $\mathbb{Q}$. Hardness results will use unary encodings of utility values (unless noted otherwise).

- PROP-CFD: Does I admit a proportional valid allocation? Find one if it exists.
- COMPLETE-EF-CFD: Does I admit a complete envy-free valid allocation? Find one if it exists.
- MMS-CFD: Does I admit an MMS allocation? Find one if it exists.

We note that, given a valid allocation, one can check in polynomial time whether it is proportional or envy-free; thus, the respective computational problems are in NP. In contrast, the problem of deciding whether a given allocation is Pareto-optimal is coNP-complete, and the problem of deciding whether there is a Pareto-optimal and envy-free allocation is Σ_2^p-complete, even when the underlying graph is a clique [44].

5.2 Maximin Share Guarantee

After Budish [31] introduced the notion of an MMS allocation, it was open for some time whether every allocation problem (without connectivity constraints) admitted such an allocation. Procaccia and Wang [82] found a counterexample. A family of more compact examples was found by Kurokawa et al. [68]; these examples implicitly use an underlying grid graph; hence, for grid graphs, existence of MMS allocations is not guaranteed. Here, we show that for *forests* an MMS allocation always exists. Our argument is constructive, and our algorithm corresponds to a discrete version of the last-diminisher method, which ensures proportionality while cutting a divisible resource [see, e.g., Brams [29]]. This method proceeds by letting one player identify a bundle of items. Every other player, in order, then has the option to *diminish* this bundle by removing some of the items from it. The last player who chose to diminish is allocated the (diminished) bundle. The same procedure is then applied to divide the rest of the cake among the remaining $n-1$ players.

We first describe an efficient procedure that guarantees each player a pre-specified level of utility.

Theorem 5.1. *Let $I = (G, N, \mathcal{U})$ be an instance of CFD where G is a forest and let $(q_i)_{i\in N}$ be an n-tuple of rational numbers. If $\mathrm{mms}_i(I) \geqslant q_i$ for all $i \in N$, then there exists a valid allocation π such that each player $i \in N$ receives the bundle of value at least q_i, i.e., $u_i(\pi(i)) \geqslant q_i$. Moreover, one can compute such an allocation in polynomial time.*

Proof. We will give an informal description of our recursive algorithm $\mathcal{A}$ (Algorithm 6), followed by pseudocode. For each $X \subseteq V$, we let $G \setminus X$ denote the subgraph induced by $V \setminus X$; also, we denote the restriction of u_i to X by $u_i|_X$.

The algorithm first checks whether its input graph G' has a value of at least q_i for each player $i \in N'$; if this is not the case, it fails. Then, if there is only one player, the algorithm simply returns the allocation that assigns all items to that player. When there are at least two players, $\mathcal{A}$ turns each connected component of the graph into a rooted tree by choosing an arbitrary node as its root; denote by $\mathrm{desc}(v)$ the set of descendants of a vertex v in this rooted tree. Then each player i finds a vertex v_i such that his value for $\mathrm{desc}(v_i)$ is at least q_i,

but for each child w of v his value for $\text{desc}(w)$ is less than q_i. The algorithm then allocates $\text{desc}(v_i)$ to the *last-diminisher* i whose vertex v_i has minimal height (such a pair (i, v_i) can be found by starting at the root of the tree and moving downwards). The player i exits with the bundle $\text{desc}(v_i)$, and the same algorithm $\mathcal{A}$ is called on the remaining instance (see Fig. 5.1).

Algorithm 6: $\mathcal{A}(I', (q_i)_{i \in N'})$

input : $I' = (G', N', \mathcal{U}')$ and $(q_i)_{i \in N'}$ where G' is a subtree of G, N' is a subset of N, and $u'_i = u_i|_{V'}$ for all $i \in N'$

output : A valid allocation π such that $u_i(\pi(i)) \geqslant q_i$ for all $i \in N'$

if $u'_i(V') < q_i$ *for some* $i \in N'$ **then**
  Return fail;
else if $|N'| = 1$ **then**
  Return π where $\pi(i) = V'$ for $\{i\} = N'$;
else
  Turn each connected component of G' into a rooted tree;
  Find $i \in N'$ and $v_i \in V'$ such that $u'_i(\text{desc}(v_i)) \geq q_i$, but $u'_j(\text{desc}(w)) < q_j$ for each child w of v_i and each player $j \in N'$;
  Set $I'' = (G' \setminus \text{desc}(v_i), N' \setminus \{i\}, \mathcal{U}'')$ where $\mathcal{U}''$ is given by $u''_j = u'_j|_{V' \setminus \text{desc}(v_i)}$ for all $j \in N' \setminus \{i\}$;
  if $\mathcal{A}(I'', (q_j)_{j \in N' \setminus \{i\}})$ *does not fail* **then**
    Set $\pi' \leftarrow \mathcal{A}(I'', (q_j)_{j \in N' \setminus \{i\}})$;
    Set $\pi(i) = \text{desc}(v_i)$ and $\pi(j) = \pi'(j)$ for each $j \in N' \setminus \{i\}$;
    Return π;

It is immediate that $\mathcal{A}$ runs in polynomial time. Let $I_n, \ldots, I_1$ be the sequence of instances constructed by $\mathcal{A}$ when called on I and $(q_i)_{i \in N}$, where $I_k = (G_k, N_k, \mathcal{U}_k)$ and $|N_k| = k$ (i.e., $I = I_n$). If $\mathcal{A}$ does not fail on any of these instances, then $\mathcal{A}(I, (q_i)_{i \in N})$ returns a desired allocation: each agent is allocated a bundle that she values at least as highly as her given value q_i. We need to prove that none of the recursive calls fails. To this end, we will prove the following lemma.

Lemma 5.1. $\text{mms}_j(I_k) \geqslant q_j$ *for all* $k \in [n]$ *and all* $j \in N_k$.

Proof. The proof proceeds by backwards induction on k. For $k = n$ the statement of the lemma is true. Suppose that the claim is true for some $k > 1$; we will prove it for $k - 1$. Consider the player $i \in N_k \setminus N_{k-1}$, and let $\text{desc}(v_i)$ be the bundle allocated to this player. For each player $j \in N_{k-1} = N_k \setminus \{i\}$ by the inductive hypothesis we have $\text{mms}_j(I_k) \geqslant q_j$. Consider a partition $\mathcal{P} = (P_1, \ldots, P_k)$ witnessing this; $u_j(P_\ell) \geqslant q_j$ for each $\ell \in [k]$.

Assume without loss of generality that $v_i \in P_1$. Then $\text{desc}(v_i)$ is fully contained in P_1: if there is a vertex w in $\text{desc}(v_i) \setminus P_1$, then the part of $\mathcal{P}$ that contains w is fully contained in a subtree rooted at a child of v_i, and hence the value of that part is strictly less than q_j, a contradiction.

Now, if $P_1 \setminus \text{desc}(v_i)$ is not empty, then it is a subtree of G, and there is another part $P \in \mathcal{P}$ that is adjacent to $P_1 \setminus \text{desc}(v_i)$ in G, i.e., there is an edge $\{w, w'\} \in E$ with $w \in P$, $w' \in P_1 \setminus \text{desc}(v)$. Therefore, $\mathcal{P}' = (\mathcal{P} \setminus \{P_1\}) \cup \{P \cup (P_1 \setminus \text{desc}(v_i))\}$ is a partition of G_{k-1} into $k-1$ connected components. By construction, $u_j(P') \geqslant q_j$ for each $P' \in \mathcal{P}'$, which proves that $\text{mms}_j(I_{k-1}) \geqslant q_j$. □

Now, consider what happens when $\mathcal{A}$ is called on I_k and $(q_i)_{i \in N_k}$ for some $k \in [n]$. Let $G_k = (V_k, E_k)$. We have $u_i(V_k) \geqslant \text{mms}_i(I_k) \geqslant q_i$ for all $i \in N_k$, which implies that the algorithm does not fail. This completes the proof. □

Theorem 5.1 relies on being given $(q_i)_{i \in N}$ as its input, so we still need to show that MMS values on forests can be computed efficiently. It turns out that this can be accomplished by the same recursive algorithm. We note that for the general problem (without graph constraints, or equivalently, on complete graphs), computing MMS values is NP-hard, though they can be well-approximated [95].

Lemma 5.2. *For an instance $I = (G, N, \mathcal{U})$ of CFD where G is a forest, and a player $i \in N$, we can compute $\text{mms}_i(I)$ in polynomial time.*

Proof. Fix a player $i \in N$. If $u_i(v)$ is represented as x_v/y_v, where x_v and y_v are integers (recall that u_i is assumed to take values in $\mathbb{Q}$), set $u'_i(v) = u_i(v) \prod_{v \in V} y_v$. Let $\text{mms}'_i(I)$ be the maximin share of player i with respect to these new utilities. Then $\text{mms}'_i(I)$ is an integer between 0 and mL^{m+1}, where $L = \max_{v \in V} \max\{x_v, y_v\}$ and

$$\text{mms}_i(I) = \frac{1}{\prod_{v \in V} y_v} \text{mms}'_i(I)$$

We now explain how to compute $\text{mms}'_i(I)$ in time polynomial in n, m, and $\log L$.

Calculating $\text{mms}_i(I)$ is the same as maximizing the worst payoff for the instance I'' where all players are copies of player i. That is, $\text{mms}'_i(I)$ is equal to the maximum positive integer of $q \leqslant mL^{m+1}$ such that $I'' = (G, N'', \mathcal{U}'')$ where N'' is a set of n copies of i and $\mathcal{U}''$ is given by $u''_j = u'_i$ for all $j \in N''$ admits a valid allocation π such that $u''_j(\pi(j)) \geqslant q$ for each copy $j \in N''$. Further, if such an allocation exists, then $\text{mms}_j(I'') \geqslant q$ for all $j \in N''$, and hence the call $\mathcal{A}(I'', (q, \ldots, q))$ of the recursive algorithm in the proof of Theorem 5.1 does not fail, returning a desired partition of G. Conversely, if $\mathcal{A}(I'', (q, \ldots, q))$ does not fail, I'' clearly admits a valid allocation where each copy of i gets a piece of value at least q.

Thus, the maximum value of such q can be found by binary search, which would require $O((m+1)\log L)$ calls to the subroutine $\mathcal{A}$ and the running time of the subroutine itself is polynomial in m and $\log L$. □

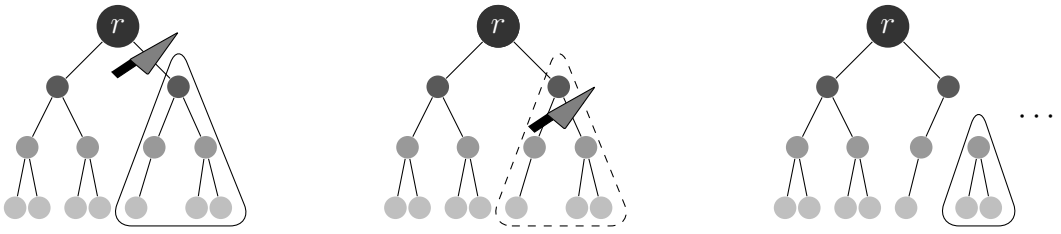


Figure 5.1: A discrete version of the last-diminisher method. (a) The first player proposes a bundle. (b) The first player proposes a bundle. (c)The last-diminisher receives the bundle.

It now follows from Theorem 5.1 and Lemma 5.2 that for forests an MMS allocation can be computed efficiently.

Theorem 5.2. *Every instance $I = (G, N, \mathcal{U})$ of CFD where G is a forest admits an MMS allocation. Moreover, such an allocation can be computed in polynomial time.*

Theorem 5.2 implies that an MMS allocation can always be found efficiently when the graph is a path, which corresponds to a linear discrete 'cake'. For this case, the algorithm presented is a discrete version of the Dubins–Spanier moving-knife procedure ensuring proportionality while cutting a continuous cake [see e.g. Brams [29]]. Our algorithm moves a knife along the path, and players shout 'cut' whenever the left part of the path has a value of at least mms_i.

Another implication of Theorem 5.2 is that on forests, there is an allocation that is both fair and efficient.

Theorem 5.3. *Every instance $I = (G, N, \mathcal{U})$ of CFD where G is a forest admits a Pareto-optimal MMS valid allocation.*

Proof. We have shown that if G is a forest, there exists an MMS allocation π. We will find a Pareto-improvement of π and update π until there is no such an improvement. It is easy to see that this procedure terminates in a finite number of steps. By construction, the resulting allocation is an MMS and Pareto-optimal. □

Nevertheless, finding such an allocation turns out to be hard for arbitrary forests. The proof is similar to the subsequent hardness proof for proportionality and can be found in Appendix A.3.

Theorem 5.4. *If one can find a Pareto-optimal valid allocation in an instance $I = (G, N, \mathcal{U})$ of CFD whose underlying graph G is a forest in polynomial time, then P = NP.*

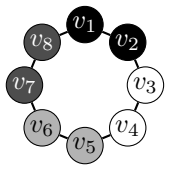
5.2.1 Non-existence of MMS Allocations on Cycles

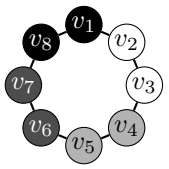
The known examples of instances without MMS allocations are very intricate. Our graph-based setting allows for simpler constructions: our next example shows that an MMS allocation may not exist on a cycle of 8 vertices. We conjecture that this is the shortest cycle that admits such an example. Our example is similar in spirit to an example for 2-additive utility functions by Bouveret and Lemaître [28].

Example 5.1. Consider an instance $I = (G, N, \mathcal{U})$ of CFD where $G = (V, E)$ with $V = \{ v_i \mid i = 1, 2, \ldots, 8 \}$, $E = \{ \{v_i, v_{i+1}\} \mid i = 1, 2, \ldots, 7 \} \cup \{\{v_1, v_8\}\}$, $N = \{1, 2, 3, 4\}$, and the utilities are given as follows.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
Players 1 & 2	1	4	4	1	3	2	2	3
Players 3 & 4	4	4	1	3	2	2	3	1

To normalize to 1, each utility value above is divided by 20. Now, we have $\mathrm{mms}_1(I) = \mathrm{mms}_2(I) \geqslant 1/4$, as witnessed by the partition $P_1 = \{\{v_1, v_2\}, \{v_3, v_4\}, \{v_5, v_6\}, \{v_7, v_8\}\}$, which offers value $1/4$ for these players. Similarly, we have $\mathrm{mms}_3(I) = \mathrm{mms}_4(I) \geqslant 1/4$, as witnessed by the partition $P_2 = \{\{v_2, v_3\}, \{v_4, v_5\}, \{v_6, v_7\}, \{v_8, v_1\}\}$. These two partitions are illustrated below (note the cyclic shift):

P_1: 

P_2: 

Now, suppose towards a contradiction that the instance I admits an MMS allocation π. Then π has to allocate at least two vertices to each player, as no player values any single item at $1/4$ or higher. This means that π partitions the cycle into either P_1 or P_2. Suppose first that π cuts the graph into P_1. Then, there is only one connected piece in P_1 that players 3 and 4 value at $1/4$ or higher, namely, $\{v_1, v_2\}$, so at least one of these players is allocated a piece whose value is less than his maximin share. A similar argument holds when π cuts the graph into P_2. Therefore, there is no MMS allocation. □

5.3 Proportionality

In this section, we shall see that the positive results in the previous section do not extend to a stronger solution concept: indeed, it is hard to find a proportional allocation, even if the graph G is a path.

Theorem 5.5. PROP-CFD *is* NP*-complete even if* G *is a path.*

Proof. Our problem is clearly in NP. We reduce from the NP-complete problem EXACT-3-COVER (X3C). The proof is similar to the one for Theorem 3 in [4]. Recall that an instance of X3C is given by a set of elements $X = \{x_1, x_2, \ldots, x_{3s}\}$ and a family $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ of three-element subsets of X; it is a 'yes'-instance if and only if X can be covered by s sets from $\mathcal{S}$. This problem remains NP-complete if for each element $x \in X$ its frequency $p_x = |\{S \in \mathcal{S} : x \in S\}|$ is at most 3.

Consider an instance $J = (X, \mathcal{S})$ of X3C; for each $S \in \mathcal{S}$, we denote the elements of S by x_S^1, x_S^2, x_S^3. We construct an instance I of PROP-CFD as follows. There are three small vertices x_S^1, x_S^2, x_S^3 for each set $S \in \mathcal{S}$, a set of s big vertices $B = \{b_1, b_2, \ldots, b_s\}$ and a dummy vertex w. The edges of G are shown in Figure 5.2.

There is one player i_S for each $S \in \mathcal{S}$, one player i_x for each $x \in X$ and one dummy player d. Hence the total number of players is $n = 3s + r + 1$. Define the utilities as:

$$u_{i_S}(v) = \begin{cases} 1/(3n) & \text{if } v = x_S^k \\ 1/n & \text{if } v \in B \\ (n-s-1)/n & \text{if } v = w \\ 0 & \text{otherwise} \end{cases}$$

$$u_{i_x}(v) = \begin{cases} 1/n & \text{if } v = x_S^k \text{ and } x \in S \\ (n-3p_x)/n & \text{if } v = w \\ 0 & \text{otherwise} \end{cases}$$

$$u_d(v) = \begin{cases} 1 & \text{if } v = w \\ 0 & \text{otherwise} \end{cases}$$

By construction $u_i(V) = 1$ for each $i \in N$. As player d assigns a positive value to vertex w only, she must receive this vertex in every proportional allocation. Given that w is allocated to d, an allocation is proportional if and only if each player i_x receives a small vertex x_S^k such that $x \in S$, and each player i_S receives vertices x_S^1, x_S^2, x_S^3 (a triple interval) or a vertex from B.

Suppose that J admits a cover $\mathcal{S}'$ of size s. Let μ be a matching between $\mathcal{S}'$ and B. Assign intervals to players i_x and i_S as follows:

- For each $S \in \mathcal{S}'$, player i_S is assigned to vertex $\mu(S)$.

- For each $S \notin \mathcal{S}'$, player i_S is assigned to the triple interval x_S^1, x_S^2, x_S^3.
- Each player i_x is assigned to the small vertex x_S^k such that $x = x_S^k$ and $S \in \mathcal{S}'$.

Then each player is assigned one connected piece and her value for that piece is at least $1/n$.

Conversely, suppose that I admits a proportional valid allocation. As $|B| = s$, the number of S-players assigned to triple intervals is $r - s$. Hence, the number of triple intervals available for x-players is s, and the respective sets constitute an exact cover for X. □

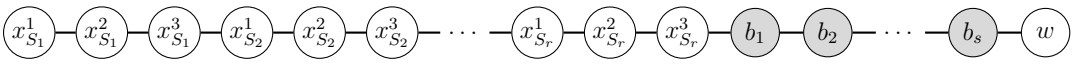


Figure 5.2: Graph constructed in the proof of Theorem 5.5.

In contrast, if G is a star, finding a proportional allocation is easy. Our algorithm for this problem, as well as all other algorithms in this section, use matching techniques and can be adapted to also find valid allocations that maximize egalitarian welfare, i.e., the utility of the worst-off agent.

Theorem 5.6. PROP-CFD *is solvable in polynomial time if G is a star.*

Proof. Let c denote the center of the star. For each player $i \in N$ we check whether there is a proportional valid allocation π assigning c to i. To this end, we create a bipartite graph $H = (Z, Z', L)$ with $Z = N \setminus \{i\}$, $Z' = V \setminus \{c\}$ and $\{j, v\} \in L$ if and only if $u_j(v) \geqslant 1/n$; the weight of this edge is $u_i(v)$. Note that $|Z| \leqslant |Z'|$. We say that a matching in H is perfect if it matches all vertices in Z. Now, observe that I admits a proportional valid allocation π that assigns c to i if and only if H admits a perfect matching M of weight $w(M) \leqslant (n-1)/n$; indeed, assigning c together with all remaining vertices to agent i gives her a connected piece that she values as $1 - w(M) \geqslant 1/n$. It remains to observe that a minimum-weight perfect matching can be computed in polynomial time [see e.g. Chapter 11 of Korte and Möhring [65]]. □

5.3.1 Bounded Number of Agent Types

If the underlying graph is a path and all players are of the same type, then a simple greedy algorithm finds a proportional allocation (or reports that none exists) in linear time: we build connected pieces one by one, by moving along the path from left to right and adding vertices to the current piece until its value to a player reaches $1/n$; at this point we start building a new piece. This procedure creates at most n pieces; a proportional valid allocation exists

if and only if it creates exactly n pieces. More generally, if G is a path and the number of agent types is bounded by a constant, a simple dynamic program can check the existence of a proportional allocation in polynomial time.

Theorem 5.7. PROP-CFD *is in* XP *with respect to the number of player types* p *if* G *is a path.*

Proof. Let $G = (V, E)$, where $V = \{v_1, \ldots, v_m\}$ and $E = \{\{v_i, v_{i+1}\} : i \in [m-1]\}$. Suppose there are n_t players of type t, for $t \in [p]$. We say that a player is *happy* if she gets a connected piece of value at least $1/n$. Let $V_0 = \emptyset$ and $V_i = \{v_1, \ldots, v_i\}$, $i > 1$.

For $i = 0, \ldots, m$, and a collection of indices $j_1, \ldots, j_p$ such that $0 \leqslant j_k \leqslant n$ for each $k \in [p]$, let $A_i[j_1, \ldots, j_p] = 1$ if there exists a valid partial allocation π of V_i with j_k happy agents of type k, $k \in [p]$, and let $A_i[j_1, \ldots, j_p] = 0$ otherwise. Clearly, $A_0[j_1, \ldots, j_p] = 1$ if and only if $j_k = 0$ for all $k \in [p]$. For $i = 1, \ldots, m$, we have $A_i[j_1, \ldots, j_p] = 1$ if and only if there exists a value $s < i$ and $t \in [p]$ such that $A_s[j_1, \ldots, j_t - 1, \ldots, j_p] = 1$ and a player of type t values the set of items $\{v_{s+1}, \ldots, v_i\}$ at $1/n$ or higher.

A proportional allocation exists if $A_m[j_1, \ldots, j_p] = 1$ for some collection of indices $j_1, \ldots, j_p$ such that $j_t \geqslant n_t$ for all $t \in [p]$. There are at most $(m+1)(n+1)^p$ values to compute, and each value can be found in time $O(mp)$ using unit cost arithmetics. Thus, PROP-CFD is in XP with respect to p. □

5.3.2 Bounded Number of Agents

If the number of agents n is bounded by a constant and G is a tree, then PROP-CFD can be solved in polynomial time: we consider all possible ways of partitioning the tree in n non-empty connected pieces (a partition Scan be associated with a set of $n-1$ edges to be deleted, so there are $\binom{m-1}{n-1}$ partitions), and, for each partition, we check if each player can be matched to a piece that she values at $1/n$ or higher (by solving a simple bipartite matching problem). This shows that PROP-CFD on trees is in XP with respect to the number of players. A more careful argument shows that PROP-CFD on trees, is, in fact, fixed parameter tractable with respect to this (weaker) parameter.

Theorem 5.8. PROP-CFD *is in* FPT *with respect to* n *when* G *is a tree.*

Proof. We turn G into a rooted tree by choosing an arbitrary node r as the root. Given a vertex v, we denote by $\mathrm{ch}(v)$ the set of children of v and by $\mathrm{desc}(v)$ the set of descendants of v (including v) in the rooted tree.

For each $v \in V$, $S \subsetneq N$, and $i \in N \setminus S$, let $\Pi_{i,v,S}$ be the set of all valid allocations $\pi : S \cup \{i\} \to 2^{\text{desc}(v)}$ with $v \in \pi(i)$ and $u_j(\pi(j)) \geqslant 1/n$ for all $j \in S$, and define

$$A_v[i, S] = \max \{u_i(\pi(i)) : \pi \in \Pi_{i,v,S}\};$$

by convention, $A_v[i, S] = -\infty$ when $\Pi_{i,v,S}$ is empty. Note that we have a 'yes'-instance of PROP-CFD if and only if $A_r[i, N \setminus \{i\}] \geqslant 1/n$ for some $i \in N$.

We will now explain how to compute all values $A_v[i, S]$ in a bottom-up manner. When v is a leaf of the rooted tree, we set $A_v[i, S] = u_i(v)$ if $S = \emptyset$ and $A_v[i, S] = -\infty$ otherwise.

Now, suppose that v is an internal vertex of the rooted tree. If $S = \emptyset$, we have$A_v[i, S] = u_i(\text{desc}(v))$, so from now on assume that $S \neq \emptyset$. We note that for each allocation $\pi \in \Pi_{i,v,S}$ the bundle of each player in S is fully contained in a subtree $\text{desc}(z)$ for some $z \in \text{ch}(v)$. This induces a partition of players in S into $|\text{ch}(v)|$ groups. To find an allocation $\pi \in \Pi_{i,v,S}$ that maximizes the utility of player i, we go through all possible partitions of S with at most $|\text{ch}(v)|$ parts. For each such partition $\mathcal{P}$, we try to find a valid allocation $\pi \in \Pi_{i,v,S}$ such that for each part $P \in \mathcal{P}$ there is a unique child z of v such that the bundle of each player $i \in P$ is fully contained in $\text{desc}(z)$; among such allocations, we pick one that maximizes $u_i(\pi(i))$.

To find such an allocation, we will construct an instance of the matching problem, which will decide which part of $\mathcal{P}$ is assigned to which $z \in \text{ch}(v)$. Thus, we construct a weighted bipartite graph $H_{\mathcal{P}} = (Z, Z', L)$ as follows. We introduce one node P for each part $P \in \mathcal{P}$ and a set X of $|\text{ch}(v)| - |\mathcal{P}|$ dummy nodes and set $Z = \mathcal{P} \cup X$. We let $Z' = \text{ch}(v)$. By construction, $|Z| = |Z'|$.

For every pair (P, z) such that $P \in \mathcal{P}$, $z \in Z'$, and players in P can be allocated items in $\text{desc}(z)$, i.e., $A_z[i, P] \neq -\infty$ or $A_z[j, P \setminus \{j\}] \geq \frac{1}{n}$ for some $j \in P$, we construct an edge $\{P, z\} \in L$ with weight $w(P, z)$ corresponding to the maximum utility that player i can receive from the set of items $\text{desc}(z)$ under the constraint that players in P obtain pieces of $\text{desc}(z)$; specifically,

$$w(P, z) = \begin{cases} A_z[i, P] & \text{if } A_z[i, P] \neq -\infty, \\ 0 & \text{otherwise.} \end{cases}$$

For every pair (x, z) with $x \in X$, $z \in Z'$, we construct an edge $\{x, z\} \in L$ with weight $w(x, z) = u_i(\text{desc}(z))$; matching x to z corresponds to assigning the items in $\text{desc}(z)$ to player i. If $H_{\mathcal{P}}$ admits a perfect matching, we set $w(\mathcal{P})$ to be the maximum weight of a perfect matching; otherwise, we set $w(\mathcal{P}) = -\infty$. Finally, we set

$$A_v[i, S] = \max\{\, w(\mathcal{P}) : \mathcal{P} \text{ is a partition of } S \wedge |\mathcal{P}| \leqslant |\text{ch}(v)| \,\}.$$

There are at most $mn2^n$ values to compute and each value can be computed in polynomial time. This shows that PROP-CFD is in FPT for trees. □

We note that placing strong constraints on the underlying graph is crucial for obtaining the easiness results in Theorems 5.7 and 5.8. This is illustrated by the following simple proposition, obtained by adapting a proof by Demko and Hill [48] for the standard setting (with no graph constraints), which shows that the XP membership with respect to the number of players/types cannot be extended to arbitrary graphs.

Proposition 5.1. *When utilities are encoded in binary,* PROP-CFD *is* NP*-complete even for* $n = 2$*,* $p = 1$*, and even if the underlying graph* G *is bipartite.*

Proof. We describe a polynomial-time reduction from PARTITION. Recall that an instance of PARTITION is given by a set of integers $J = \{a_i : i \in H\}$ such that $\sum_{i \in H} a_i = 2k$. It is a 'yes'-instance if and only if there exists a subset of indices $H' \subset H$ such that $\sum_{i \in H'} a_i = \sum_{i \in H \setminus H'} a_i = k$.

Define an instance I of PROP-CFD as follows. Let $G = (V, E)$ where $V = \{v_i : i \in H\} \cup \{w_1, w_2\}$ and $E = \{\{v_i, w_1\}, \{v_i, w_2\} : i \in H\}$. There are two players with the same utility function $u(v_i) = a_i/(2k)$ for $i \in H$ and $u(w_1) = u(w_2) = 0$. Then I admits a proportional valid allocation if and only if J is a 'yes'-instance of PARTITION. □

5.4 Envy-freeness

Envy-freeness turns out to be computationally more challenging than proportionality: finding a complete envy-free allocation is NP-hard even if the underlying graph is a star (for complete graphs, this result is shown by Lipton et al. [73]).

Theorem 5.9. COMPLETE-EF-CFD *is* NP*-complete even if* G *is a star.*

Proof. Our hardness proof proceeds by a reduction from INDEPENDENT SET. Recall that an instance of INDEPENDENT SET is given by an undirected graph (W, L) and an integer k; it is a 'yes'-instance if and only if (W, L) contains an independent set of size k. Given an instance (W, L) of INDEPENDENT SET, we construct an instance of COMPLETE-EF-CFD as follows. For each vertex $w \in W$ we create an item w and a player i_w. Similarly, for each edge $\ell \in L$ we create an item ℓ and a player i_ℓ. We also create a set of dummy items Z with $|Z| = k$, as well as an item c and a player i_c. The graph G is a star with center c and set of leaves $W \cup L \cup Z$. Finally, define utility functions as follows.

- For each $w \in W$, we set $u_{i_w}(w) = 1/(k+1)$ and $u_{i_w}(z) = 1/(k+1)$ for each $z \in Z$.
- For each $\ell \in L$ with $\ell = \{x, y\}$, we set $u_{i_\ell}(\ell) = 3/7$, $u_{i_\ell}(x) = u_{i_\ell}(y) = 2/7$.
- We set $u_{i_c}(c) = 1$.

- All other utilities are set to 0.

We will now argue that there exists an independent set of size k in the graph (W, L) if and only if this instance of CFD admits a complete envy-free valid allocation.

Suppose there exists an independent set $X \subseteq W$ of size k. We construct an allocation π as follows:

- player i_c receives $X \cup \{c\}$;
- for $w \in W \setminus X$, player i_w receives w;
- for $w \in X$, player i_w receives one item in Z;
- for $\ell \in L$, player i_ℓ receives ℓ.

Clearly, π is a complete valid allocation. It remains to show that π is envy-free. First, player i_c does not envy any other player since she receives all her positive-utility items. Vertex players $\{i_w : w \in W\}$ receive utility $1/(k+1)$ in π; they could only envy someone who has multiple dummies, but no one does. Edge players $\{i_\ell : \ell \in L\}$ receive utility $3/7$ in π; the only way an edge player i_ℓ could envy another player is if that player got both items corresponding to endpoints of ℓ. But the only player who receives more than one vertex item is player i_c whose items correspond to an independent set. So no player envies anyone, and π is envy-free.

Conversely, suppose that there is a complete envy-free valid allocation π. By completeness, π allocates the central piece c to some player. If i_c does not receive c then she would envy the player who receives it; so $c \in \pi(i_c)$. Thus, every other player receives at most one item. Since π is complete, this means that i_c gets at least k leaf items. Further, if i_ℓ does not receive ℓ, she would envy the player who receives it, so $\pi(i_\ell) = \{\ell\}$. Next, consider the bundle of player i_c. If it contains more than one dummy item, vertex players would envy i_c. Thus, it contains at least one item $w \in W$. If $\pi(i_c)$ also contains a dummy item, i_w would envy i_c, so $\pi(i_c)$ consists of c and k vertex items. Now, if there is an edge $\ell = (x, y)$ such that $x, y \in \pi(i_c)$, then player i_ℓ envies i_c. Hence, $\pi(i_c) \setminus \{c\}$ forms an independent set of size k in (W, L). □

We also obtain a hardness result for paths; the proof is similar to that of Theorem 5.5.

Theorem 5.10. *The problem* COMPLETE-EF-CFD *is* NP*-complete even if G is a path.*

Proof. We shall show how to modify the polynomial transformation from X3C provided in the proof of Theorem 5.5.

Consider an instance $J = (X, \mathcal{S})$ of X3C where for each element $x \in X$ its frequency $p_x = |\{ S \in \mathcal{S} \mid x \in S \}|$ is at most 3. Recall that an instance of X3C is given by a set of elements $X = \{x_1, x_2, \ldots, x_{3s}\}$ and a family $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ of three-element subsets of X; for each $S \in \mathcal{S}$, we denote the elements of S by x_S^1, x_S^2, x_S^3.

We create the same graph G as in the proof of Theorem 5.5 and to the right end of the graph, we attach a path that consists of $s+1$ dummy vertices $w_1, w_2, \ldots, w_{s+1}$. There is one player i_S for each $S \in \mathcal{S}$, one player i_x for each $x \in X$ and $s+1$ dummy players $d_1, d_2, \ldots, d_{s+1}$. Hence the total number of players is $n = 4s + r + 1$.

For each x, we let W_x be an arbitrary set of $s + 1 - p_x$ vertices among the dummy vertices. We define the utilities as:

$$u_{i_S}(v) = \begin{cases} 1/(3s+3) & \text{if } v = x_S^k \\ 1/(s+1) & \text{if } v \in B \\ 0 & \text{otherwise} \end{cases}$$

$$u_{i_x}(v) = \begin{cases} 1/(s+1) & \text{if } v = x_S^k \text{ and } x \in S \\ 1/(s+1) & \text{if } v \in W_x \\ 0 & \text{otherwise} \end{cases}$$

$$u_{d_k}(v) = \begin{cases} 1 & \text{if } v = w_k \\ 0 & \text{otherwise} \end{cases}$$

By construction $u_i(V) = 1$ for each $i \in N$. As each dummy player d_k assigns a positive value to the dummy vertex w_k only, she must receive this vertex in every envy-free complete allocation. Given that each w_k is allocated to d_k, each non-dummy player should receive a bundle of value at least $1/(s+1)$ in every envy-free complete allocation. Namely, if a player i_x receives a bundle of value less than $1/(s+1)$ then she will envy any dummy player; if a player i_S receives a bundle of value less than $1/(s+1)$ then she will envy any player allocated to a good in B.

Suppose that J admits a cover $\mathcal{S}'$ of size s. Let μ be a matching between $\mathcal{S}'$ and B. Assign intervals to players i_x and i_S as follows:

- For each $S \in \mathcal{S}'$, player i_S is assigned to vertex $\mu(S)$.
- For each $S \notin \mathcal{S}'$, player i_S is assigned to the triple interval x_S^1, x_S^2, x_S^3.
- Each player i_x is assigned to the small vertex x_S^k such that $x = x_S^k$ and $S \in \mathcal{S}'$.

Assign each dummy vertex w_k to the corresponding dummy player d_k. Then the resulting allocation is clearly a complete envy-free valid allocation.

Conversely, suppose that a complete envy-free valid allocation in I exists. As we have seen earlier, each player should receive at least $1/(s+1)$, which is possible only if each

player i_S either receives one from B or the triple interval x_S^1, x_S^2, x_S^3. As the number of vertices in B is only s, the number of players i_S assigned to triple intervals is $r - s$. So the number of S-vertices available for players i_x is $3s$ and they constitute an exact cover for J. □

On the positive side, just as for PROP-CFD, the problem COMPLETE-EF-CFD is also in XP with respect to the number of player types p, as long as G is a path.

Theorem 5.11. COMPLETE-EF-CFD *is in* XP *with respect to the number of player types* p *if* G *is a path.*

Proof. Note that for an allocation to be envy-free, all pieces assigned to players of a given type should have the same value to players of that type. When G is a path, there are only $\binom{m+1}{2} \leqslant m^2$ different connected bundles. Hence there are at most m^2 many possibilities for the utility that a player of a given type can obtain in a valid allocation.

Our algorithm works as follows. For each player type, it guesses the utility that players of that type assign to their pieces (this guessing can be implemented by going over all possibilities, as there are at most $(m^2)^p$ of them). It then proceeds similarly to the dynamic programming algorithm in the proof of Theorem 5.7; the only difference is that, when creating a piece of the form $\{v_{s+1}, \ldots, v_i\}$ for a player of a given type, it checks that the utility of that player type for this piece is what it guessed for that type, and that other players' utility for this piece is at most their guessed utility.

Let $G = (V, E)$, where $V = \{v_1, \ldots, v_m\}$ and $E = \{\{v_i, v_{i+1}\} : i \in [m-1]\}$. Suppose there are n_t players of type t, for $t \in [p]$. Let $V_0 = \emptyset$ and $V_i = \{v_1, \ldots, v_i\}$, $i > 1$. Let X_k denote the set of possible utility values of type k agent; each $|X_k|$ is at most m^2 as stated above. We now provide a formal description of our algorithm as follows.

For each $(x_1, x_2, \ldots, x_p) \in \prod_{k=1}^{p} X_k$, $i = 0, \ldots, m$, and a collection of indices $j_1, \ldots, j_p$ such that $0 \leqslant j_k \leqslant n$ for each $k \in [p]$, let $A_i[x_1, x_2, \ldots, x_p, j_1, \ldots, j_p] = 1$ if there exists a valid partial allocation π of $V_i = \{v_1, v_2, \ldots, v_i\}$ where there are j_k agents of each type $k \in [p]$ receiving a bundle of value exactly x_k and no agent who has been assigned to a bundle envies the other's bundle, and let $A_i[x_1, x_2, \ldots, x_p, j_1, \ldots, j_p] = 0$ otherwise. Clearly, $A_0[x_1, x_2, \ldots, x_p, j_1, \ldots, j_p] = 1$ if and only if $x_k = 0$ and $j_k = 0$ for all $k \in [p]$. For $i = 1, \ldots, m$, we have $A_i[x_1, x_2, \ldots, x_p, j_1, \ldots, j_p] = 1$ if and only if there exists a value $s < i$ and $t \in [p]$ such that $A_s[x_1, x_2, \ldots, x_p, j_1, \ldots, j_t - 1, \ldots, j_p] = 1$ and a player of type t values the set of items $\{v_{s+1}, \ldots, v_i\}$ at exactly x_t and no player of type $k \neq t$ values the items $\{v_{s+1}, \ldots, v_i\}$ strictly higher than x_k.

An envy-free complete allocation exists if $A_m[x_1, x_2, \ldots, x_p, n_1, \ldots, n_p] = 1$ for some collection of utilities $(x_1, \ldots, x_p) \in \prod_{k=1}^{p} X_k$. There are at most $m^{2p}(n+1)^p$ values to

compute, and each value can be found in time $O(mp)$ using unit cost arithmetics. Thus, PROP-CFD is in XP with respect to p. □

5.5 Summary and Related Work

We studied the problem of fairly allocating indivisible goods under the constraint that each bundle is connected in the given graph. We summarize our results in Table 5.1.

There are several exciting directions for the study of connected fair division of indivisible goods. For the solution concepts we have studied in this paper, one can ask whether certain graph classes yield better approximations than the general case, both in terms of existence guarantees and complexity results. In particular, it would be interesting to obtain a characterization of graphs for which an MMS allocation is guaranteed to exist: we have seen that on forests, MMS allocations always exists, but the same is not true for large cycles and grid graphs. It would be interesting to move towards a characterization of graph classes which guarantee existence. The existing results may suggest that graphs with low circumference are particularly promising.

There are also further solution concepts that we have not considered, most notably the maximum Nash welfare solution [see Caragiannis et al. [33]], which could be studied in this context both from the axiomatic and the computational points of view. Another promising direction would be to extend the work to other preference representations, including ordinal preferences Aziz et al. [8], or to chores instead of goods [e.g., Aziz et al. [9]]. Also, it would be interesting to obtain analogues of procedures such as sequential allocation and round-robin that respect the connectivity constraints and still produce desirable allocations.

Finally, we may consider placing constraints on the 'shapes' of players' pieces, e.g., by requiring that the size of each piece is large relative to its diameter; similar ideas have been recently explored by Segal-Halevi et al. [85] in the context of the land division problem (i.e., cutting a 2-dimensional cake).

Related work In the context of fair allocation of *divisible* items (also known as cake-cutting) contiguity is a well-studied requirement. For instance, Stromquist [89] showed that an envy-free division in which each player receives a single contiguous piece always exists, but it cannot be obtained by a finite algorithm, even for three players [90]. These results extend to equitable division with contiguous pieces [3, 34, 35]. [17] consider fair allocations with contiguous pieces that approximately maximize social welfare; Aumann et al. [4] investigate a variant of this question without fairness constraints.

Conitzer et al. [38] analyze a combinatorial auction setting that is somewhat similar to this work: in their model, there is an undirected graph describing connections between items,

		general case	few types (p)	few players (n)
MMS	cliques			
	forests	polytime (Th. 5.2)	polytime	polytime
Proportionality	cliques	NP-c. ([73, 22])	NP-c.	
	trees	NP-c.		FPT (Th. 5.8)
	stars	polytime (Th. 5.6)	polytime	polytime
	paths	NP-c. (Th. 5.5)	XP (Th. 5.7)	FPT
Complete Envy-freeness	cliques	NP-c. ([73, 22])		
	trees	NP-c.		
	stars	NP-c. (Th. 5.9)		
	paths	NP-c. (Th. 5.10)	XP (Th. 5.11)	XP

Table 5.1: Overview of complexity results. When no reference is given, the result follows trivially from other results in the table.

and each agent's bid is connected with respect to this graph. They provide an algorithm for finding an allocation that maximizes the social welfare and is in FPT with respect to the treewidth of the item graph. Aumann et al. [5] consider auctioning of a time interval, and obtain results both for the case of pre-determined time slots (which corresponds to the model of Conitzer et al. [38], with the item graph being a line) and for the case where the interval can be cut into arbitrary slots (which is similar in spirit to cake-cutting). However, neither paper considers any fairness constraints.

Three recent papers combine graphs and fair division. Chevaleyre et al. [37] consider the setting where agents are located in vertices of a graph. Each agent has an initial endowment of goods and can trade with her neighbors in the graph. The authors ask what outcomes can be achieved by a sequence of mutually beneficial deals. In the works of Abebe et al. [1] and Bei et al. [18], the graph describes a visibility relation: agents are located in vertices and an agent can only envy agents who are adjacent to her. In contrast, in our model graphs represent the relationship between items rather than agents.

Part IV

Perspectives

Chapter 6

Perspectives

Incorporating the aspects of network structure helps us understand what leads to desirable outcomes and what makes them hard to achieve. Putting it simply, we showed that if the underlying network is very well-structured, i.e., forest, desirable outcomes are often guaranteed to exist and some of the problems known to be computationally hard for the unrestricted setting become efficiently solvable. This result seems to imply a natural tradeoff: we have assumed very little knowledge about underlying player preferences, and thus required a lot of structure; much of the literature makes no assumptions on the underlying network, but assumes a more restricted player preference model.

Our negative results with respect to cycles show that beyond acyclic graph structures, it is difficult to provide any theoretical guarantees, both in terms of existence and computational tractability. That said, the unconditional worst-case analysis may be too strong to provide any meaningful insights for practical applications. Largely missing at the moment is perhaps an empirical research on real-life networks. For instance, it would be very interesting to analyze what kind of deviations can occur in real-life networks; different academic communities may have different collaboration culture. One could ask the question "which deviation criterion explains the stability of some community structures."

Throughout, we have focused on whether a single normative property, such as fairness and stability, can be satisfied under certain conditions on the inputs. However, what is important in practice is to see the compatibility between multiple properties. In particular, identifying a class of coalition formation games where Pareto-optimality and stability can be simultaneously achieved would be an important topic of research.

Finally, while we have demonstrated the power of the graph-restricted formulation, there are other constraints that are prevalent in practice. A promising direction would be to study coalition formation games with diversity constraints; for example, in university class room activities, a certain balance of the members in each group is often required, along a number of dimensions.

References

[1] R. Abebe, J. M. Kleinberg, and D. C. Parkes. Fair division via social comparison. In *Proceedings of the 16th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 281–289, 2017.

[2] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[3] Y. Aumann and Y. Dombb. The efficiency of fair division with connected pieces. *ACM Transactions on Economics and Computation*, 3:23:1–23:16, 2015.

[4] Y. Aumann, Y. Dombb, and A. Hassidim. Computing socially-efficient cake divisions. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 343–350, 2013.

[5] Y. Aumann, Y. Dombb, and A. Hassidim. Auctioning time: Truthful auctions of heterogeneous divisible goods. *ACM Transactions on Economics and Computation*, 4(1):3:1–3:16, 2015.

[6] H. Aziz, F. Brandt, and H. G. Seedig. Computing desirable partitions in additively separable hedonic games. *Artificial Intelligence*, 195:316–334, Feb. 2013.

[7] H. Aziz and B. De Keijzer. Complexity of coalition structure generation. In *Proceedings of the 10th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 191–198, 2011.

[8] H. Aziz, S. Gaspers, S. Mackenzie, and T. Walsh. Fair assignment of indivisible objects under ordinal preferences. *Artificial Intelligence*, 227:71–92, 2015.

[9] H. Aziz, G. Rauchecker, G. Schryen, and T. Walsh. Algorithms for max-min share fair allocation of indivisible chores. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 335–341, 2017.

[10] H. Aziz and R. Savani. Hedonic games. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 15. Cambridge University Press, 2016.

[11] M. Balcan, F. Constantin, S. Iwata, and L. Wang. Learning valuation functions. In *Proceedings of the 25th Conference on Computational Learning Theory (COLT)*, pages 4.1–4.24, 2012.

[12] M. Balcan, A. Procaccia, and Y. Zick. Learning cooperative games. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 475–481, 2015.

[13] M. Balcan, T. Sandholm, and E. Vitercik. Sample complexity of automated mechanism design. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2083–2091, 2016.

[14] M. Balcan, E. Vitercik, and C. White. Learning combinatorial functions from pairwise comparisons. In *Proceedings of the 29th Conference on Computational Learning Theory (COLT)*, pages 1–35, 2016.

[15] C. Ballester. NP-competeness in hedonic games. *Games and Economic Behavior*, 49:1–30, 2004.

[16] S. Banerjee, H. Konishi, and T. Sönmez. Core in a simple coalition formation game. *Social Choice and Welfare*, 18(1):135–153, 2001.

[17] X. Bei, N. Chen, X. Hua, B. Tao, and E. Yang. Optimal proportional cake cutting with connected pieces. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1263–1269, 2012.

[18] X. Bei, Y. Qiao, and S. Zhang. Networked fairness in cake cutting. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3632–3638, 2017.

[19] P. Berman, M. Karpiński, and A. D. Scott. Approximation hardness for short symmetric instances of MAX-3SAT, 2003.

[20] V. Bilò, A. Fanelli, M. Flammini, G. Monaco, and L. Moscardelli. Nash stability in fractional hedonic games. In *Proceedings of the 10th Conference on Web and Internet Economics (WINE)*, pages 486–491, 2014.

[21] V. Bilò, A. Fanelli, M. Flammini, G. Monaco, and L. Moscardelli. On the price of stability of fractional hedonic games. In *Proceedings of the 14th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1239–1247, 2015.

[22] B. Bliem, R. Bredereck, and R. Niedermeier. Complexity of efficient and envy-free resource allocation: Few agents, resources, or utility levels. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 102–108, 2016.

[23] A. Bogomolnaia and M. Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.

[24] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335 – 379, 1976.

[25] N. Bousquet, Z. Li, and A. Vetta. Coalition games on interaction graphs: A horticultural perspective. In *Proceedings of the 16th ACM Conference on Economics and Computation (EC)*, pages 95–112, 2015.

[26] S. Bouveret, K. Cechlárová, E. Elkind, A. Igarashi, and D. Peters. Fair division of a graph. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 135 – 141, 2017.

[27] S. Bouveret, Y. Chevaleyre, and N. Maudet. Fair allocation of indivisible goods. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 12. Cambridge University Press, 2016.

[28] S. Bouveret and M. Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems*, 30(2):259–290, 2015.

[29] S. J. Brams and A. D. Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996.

[30] S. Brânzei, Y. Lv, and R. Mehta. To give or not to give: Fair division for single minded valuations. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 123–129, 2016.

[31] E. Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.

[32] N. Burani and W. Zwicker. Coalition formation games with separable preferences. *Mathematical Social Sciences*, 45(1):27–52, 2003.

[33] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang. The unreasonable fairness of maximum nash welfare. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, pages 305–322, 2016.

[34] K. Cechlárová, J. Doboš, and E. Pillárová. On the existence of equitable divisions. *Information Sciences*, 228:239–245, 2013.

[35] K. Cechlárová and E. Pillárová. On the computability of equitable divisions. *Discrete Optimization*, 9:249–257, 2012.

[36] G. Chalkiadakis, G. Greco, and E. Markakis. Characteristic function games with restricted agent interactions: Core-stability and coalition structures. *Artificial Intelligence*, 232:76–113, 2016.

[37] Y. Chevaleyre, U. Endriss, and N. Maudet. Distributed fair allocation of indivisible goods. *Artificial Intelligence*, 242:1–22, 2017.

[38] V. Conitzer, J. Derryberry, and T. Sandholm. Combinatorial auctions with structured item graphs. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence (AAAI)*, pages 212–218, 2004.

[39] D. G. Corneil, S. Olariu, and L. Stewart. The ultimate interval graph recognition algorithm? In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 175–180, 1998.

[40] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[41] A. Darmann. Group activity selection from ordinal preferences. In *Proceedings of the 4th International Conference on Algorithmic Decision Theory (ADT)*, pages 35–51, 2015.

[42] A. Darmann, E. Elkind, S. Kurz, J. Lang, J. Schauer, and G. Woeginger. Group activity selection problem. *International Journal of Game Theory*, 2017.

[43] A. Darmann and J. Lang. Group activity selection problems. In U. Endriss, editor, *Trends in Computational Social Choice*, chapter 5. AI Access, 2017.

[44] B. de Keijzer, S. Bouveret, T. Klos, and Y. Zhang. On the complexity of efficiency and envy-freeness in fair division of indivisible goods with additive preferences. In *Proceedings of the 1st International Conference on Algorithmic Decision Theory (ADT)*, pages 98–110, 2009.

[45] G. Demange. Intermediate preferences and stable coalition structures. *Journal of Mathematical Economics*, 23(1):45–58, 1994.

[46] G. Demange. On group stability in hierarchies and networks. *Journal of Political Economy*, 112(4):754–778, 2004.

[47] M. Demange and T. Ekim. Minimum maximal matching is NP-hard in regular bipartite graphs. In *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation (TAMC)*, pages 364–374, 2008.

[48] S. Demko and T. P. Hill. Equitable distribution of indivisible items. *Mathematical Social Sciences*, 16:145–158, 1998.

[49] D. Dimitrov, P. Borm, R. Hendrickx, and S. C. Sung. Simple priorities and core stability in hedonic games. *Social Choice and Welfare*, 26(2):421–433, 2006.

[50] M. Dom. Algorithmic aspects of the consecutive-ones property. *Bulletin of the European Association for Theoretical Computer Science*, 98:27–59, 2009.

[51] J. H. Drèze and J. Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica*, 48(4):987–1003, 1980.

[52] E. Elkind. Coalitional games on sparse social networks. In *Proceedings of the 10th Conference on Web and Internet Economics (WINE)*, pages 308–321, 2014.

[53] J. Fearnley, M. Gairing, P. Goldberg, and R. Savani. Learning equilibria of games via payoff queries. In *Proceedings of the 14th ACM Conference on Economics and Computation (EC)*, pages 397–414, 2013.

[54] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965.

[55] M. Gairing and R. Savani. Computing stable outcomes in hedonic games. In *Proceedings of the 3rd International Symposium on Algorithmic Game Theory (SAGT)*, pages 174–185, 2010.

[56] M. Gairing and R. Savani. Computing stable outcomes in hedonic games with voting-based deviations. In *Proceedings of the 10th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 559–566, 2011.

[57] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[58] S. Gupta, S. Roy, S. Saurabh, and M. Zehavi. Parameterized analysis for the group activity selection problem on graphs. In *Proceedings of the 10th International Symposium on Algorithmic Game Theory (SAGT)*, Lecture Notes in Computer Science, pages 106–118, 2017.

[59] M. Habib, R. McConnell, C. Paul, and L. Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1):59 – 84, 2000.

[60] W.-L. Hsu and T.-H. Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM Journal on Computing*, 28(3):1004–1020, 1999.

[61] A. Igarashi, R. Bredereck, and E. Elkind. On parameterized complexity of group activity selection problems on social networks. In *Proceedings of the 16th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1575 – 1577, 2017. Extended version available as arXiv:1703.01121 [cs.GT].

[62] A. Igarashi and E. Elkind. Hedonic games with graph-restricted communication. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 242–250, 2016.

[63] A. Igarashi, D. Peters, and E. Elkind. Group activity selection on social networks. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 565–571, 2017. Extended version available as arXiv:1611.04524 [cs.GT].

[64] D. S. Johnson and C. H. Papadimitriou. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[65] B. Korte and J. Vygen. *Combinatorial Optimization - Theory and Algorithms*. Springer-Verlag Berlin Heidelberg, 2006.

[66] N. Korte and R. H. Möhring. A simple linear-time algorithm to recognize interval graphs. In *Proceedings of the 2nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, Lecture Notes in Computer Science, pages 1–16, 1987.

[67] D. Kratsch, R. M. McConnell, K. Mehlhorn, and J. P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM Journal on Computing*, 36(2):326–353, 2006.

[68] D. Kurokawa, A. D. Procaccia, and J. Wang. When can the maximin share guarantee be guaranteed? In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pages 523–529, 2016.

[69] V. B. Le and F. Pfender. Complexity results for rainbow matchings. *Theoretical Computer Science*, 524(C):27–33, 2014.

[70] M. Le Breton, G. Owen, and S. Weber. Strongly balanced cooperative games. *International Journal of Game Theory*, 20(4):419–427, 1992.

[71] H. Lee and Y. Shoham. Stable invitations. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 965–971, 2015.

[72] H. Lee and V. V. Williams. Parameterized complexity of group activity selection. In *Proceedings of the 16th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 353–361, 2017.

[73] R. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Economics and Computation (EC)*, pages 125–131, 2004.

[74] R. Meir, Y. Zick, E. Elkind, and J. Rosenschein. Bounding the cost of stability in games over interaction networks. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, pages 690–696, 2013.

[75] J. Morgenstern and T. Roughgarden. On the pseudo-dimension of nearly optimal auctions. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 136–144, 2015.

[76] J. Morgenstern and T. Roughgarden. Learning simple auctions. In *Proceedings of the 29th Conference on Computational Learning Theory (COLT)*, pages 1298–1318, 2016.

[77] O. Morgenstern and J. von Neumann. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[78] R. B. Myerson. Graphs and cooperation in games. *Mathematics of Operations Research*, 2(3):225–229, 1977.

[79] J. B. Orlin. Max flows in O(nm) time, or better. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 765–774, 2013.

[80] D. Peters. Graphical hedonic games of bounded treewidth. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pages 586–593, 2016.

[81] D. Peters and E. Elkind. Simple causes of complexity in hedonic games. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 617–623, 2015.

[82] A. D. Procaccia and J. Wang. Fair enough: Guaranteeing approximate maximin shares. In *Proceedings of the 15th ACM Conference on Economics and Computation (EC)*, pages 675–692, 2014.

[83] E. Ron. NP-complete stable matching problems. *Journal of Algorithms*, 11(2):285–304, 1990.

[84] A. A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991.

[85] E. Segal-Halevi, A. Hassidim, and Y. Aumann. Envy-free cake-cutting in two dimensions. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1021–1028, 2015.

[86] T. Shrot, Y. Aumann, and S. Kraus. On agent types in coalition formation problems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 757–764, 2010.

[87] A. Sinha, D. Kar, and M. Tambe. Learning adversary behavior in security games: A PAC model perspective. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 214–222, 2016.

[88] J. Sliwinski and Y. Zick. Learning hedonic games. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2730–2736, 2017.

[89] W. Stromquist. How to cut a cake fairly. *American Mathematical Monthly*, 87:640–644, 1980.

[90] W. Stromquist. Envy-free divisions cannot be found by finite protocols. *The Electronic Journal of Combinatorics*, 15:145–158, 2008.

[91] W. Suksompong. Fairly allocating contiguous blocks of indivisible items. In *Proceedings of the 10th International Symposium on Algorithmic Game Theory (SAGT)*, Lecture Notes in Computer Science, pages 333–344, 2017. Extended version available as arXiv:1707.00345 [cs.GT].

[92] S. Sung and D. Dimitrov. Computational complexity in additive hedonic games. *European Journal of Operational Research*, 203(3):635–639, 2010.

[93] S. C. Sung and D. Dimitrov. On core membership testing for hedonic coalition formation games. *Operations Research Letters*, 35(2):155–158, 2007.

[94] N. Talmon. Structured proportional representation. In *Proceedings of the 16th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 633–641, 2017.

[95] G. J. Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154, 1997.

[96] G. J. Woeginger. *Core stability in hedonic coalition formation*, volume 7741 LNCS, pages 33–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

Appendix A

Appendix

A.1 Appendix: Chapter 2

A.1.1 Empty Instance of an IR-INS Partition

Example A.1. *Consider an additively hedonic game* (N, U, L) *consisting of five parties aligned on a path, where* $N = \{1, 2, 3, 4, 5\}$*,* $L = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}\}$*, and the utility matrix* U *is given by*

$$
\begin{aligned}
&U(1,2) = -1, U(1,3) = 2, U(1,4) = -2, U(1,5) = 0,\\
&U(2,1) = 0, U(2,3) = 0, U(2,4) = -2, U(2,5) = -2,\\
&U(3,1) = -2, U(3,2) = 2, U(3,4) = 2, U(3,5) = -2,\\
&U(4,1) = -2, U(4,2) = -2, U(4,3) = 0, U(4,5) = 0,\\
&U(5,1) = 0, U(5,2) = -2, U(5,3) = 2, U(5,4) = -1.
\end{aligned}
$$

See Figure A.1 for an illustration. Intuitively, the middle player 3 *likes his neighbours but hates the leaves; the players* 2 *and* 4 *hates the players on the other side; and the leaves* 1 *and* 5 *likes the middle player but hates his neighbor. The resulting preference profile is as follows:*

$$
\begin{aligned}
&1 \,:\, \{1,2,3\} \succ \{1\},\\
&2 \,:\, \{1,2,3\} \sim \{1,2\} \sim \{2,3\} \sim \{2\},\\
&3 \,:\, \{2,3,4\} \succ \{1,2,3,4\} \sim \{2,3,4,5\} \sim \{2,3\} \sim \{3,4\} \succ\\
&\{1,2,3,4,5\} \sim \{1,2,3\} \sim \{3,4,5\} \sim \{3\},\\
&4 \,:\, \{3,4,5\} \sim \{3,4\} \sim \{4,5\} \sim \{4\},\\
&5 \,:\, \{3,4,5\} \succ \{5\},
\end{aligned}
$$

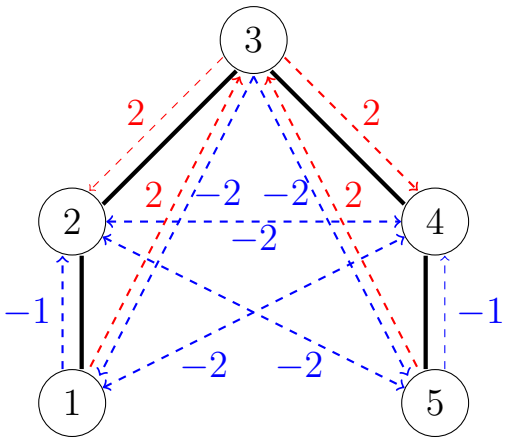


Figure A.1: Thick black lines represent communication links between players. Values on dashed lines are utilities of players with red and blue lines representing friends and enemies, respectively.

where we omit coalitions that are not individually rational or not feasible. There are four individually rational feasible partitions of this game: $\pi_1 = \{\{1\},\{2,3\},\{4,5\}\}$, $\pi_2 = \{\{1,2,3\},\{4\},\{5\}\}$, $\pi_3 = \{\{1\},\{2\},\{3,4\},\{5\}\}$, *and* $\pi_4 = \{\{1\},\{2\},\{3,4,5\}\}$. *We will show that each of them admits an IR-in-neighbor feasible deviation. Indeed, player* 1 *has an IR-in-neighbor feasible deviation to coalition* $\{2,3\}$ *in* π_1 *whereas player* 3 *has an IR-in-neighbor feasible deviation to coalition* $\{4\}$ *in* π_2. *Similarly, player* 5 *has an IR-in-neighbor feasible deviation to coalition* $\{3,4\}$ *in* π_3 *whereas player* 3 *has an IR-in-neighbor feasible deviation to coalition* $\{2\}$ *in* π_4. *Note however that both* π_1 *and* π_3 *are individually stable.*

A.1.2 Proof of Theorem 2.9

Proof. IR-in-neighbor stability can be verified in polynomial time, so our problem is in NP. We provide a reduction from CLIQUE to prove hardness.

Given an undirected graph $G = (V, E)$ and a positive integer k, we create a vertex player v for each $v \in V$ and five additional players a, b, c, d, and e; we then create a star with the center c, i.e., $N = V \cup \{a, b, c, d, e\}$, and $L = \{\,\{c, v\} \mid v \in V\,\} \cup \{\{a, c\}, \{b, c\}, \{d, c\}, \{e, c\}\}$.

The five players $\{a, b, c, d, e\}$ will form an empty IR-INS instance described in Example

A.1 as specified below.

$$
\begin{aligned}
&U(a,b) = -1, U(a,c) = 2, U(a,d) = -2, U(a,e) = 0,\\
&U(b,a) = 0, U(b,c) = 0, U(b,d) = -2, U(b,e) = -2,\\
&U(c,a) = -k, U(c,b) = k, U(c,d) = k, U(c,e) = -k,\\
&U(d,a) = -2, U(d,b) = -2, U(d,c) = 0, U(d,e) = 0,\\
&U(e,a) = 0, U(e,b) = -2, U(e,c) = 2, U(e,d) = -1.
\end{aligned}
$$

The four players a, b, d, and e dislike any vertex player while the center player c has a utility 1 for such a player. Formally, we let $M = |N| + 1$ and define utilities for these players as follows.

$$
\begin{aligned}
&U(a,v) = U(b,v) = U(d,v) = U(e,v) = -M \text{ for each } v \in V,\\
&U(c,v) = 1 \text{ for each } v \in V,
\end{aligned}
$$

Each vertex player dislikes players a, b, d and e, as well as their non-neigbhours. Specifically, the utilities of each vertex $v \in V$ are given as follows.

$$
\begin{aligned}
&U(v,a) = U(v,b) = U(v,d) = U(v,e) = -M, U(v,b) = 0\\
&U(v,u) = 0 \text{ if } \{u,v\} \in E \text{ and } U(v,u) = -M \text{ otherwise, for all } u, v \in V.
\end{aligned}
$$

We will now show that G contains a clique of size k if and only if the game admits an in-neighbor stable feasible partition.

Suppose first that G contains a clique C of size k. Let $\pi = \{\{a\}, \{b\}, \{d\}, \{e\}, C \cup \{c\}\} \cup \{\{v\} \mid v \in V \setminus C\}$. Clearly, players a, b, d and e do not want to deviate to another coalition. Also, no player $v \in V$ can profitably deviate. Player c does not want to deviate to $\{b\}$, $\{e\}$, or any singleton in $V \setminus C$ because $\sum_{v \in C} U(c,v) \geq k \geq 1$. Thus, π is IR-in-neighbor stable.

Conversely, suppose that π is an IR-in-neighbor stable feasible partition of N. Observe that no player from $\{a, b, d, e\}$ forms a coalition with vertex players due to individual rationality. Hence if the center c forms a coalition with some player in $\{a, b, d, e\}$, then $\pi(c)$ does not include any vertex player; however there is no such an IR-in-neighbor stable feasible partition as we have seen in Example A.1. Thus, in π player c is grouped together with players in V only; moreover, all players a, b, d, and e form a coalition on their own, i.e., $\{a\}, \{b\}, \{d\}, \{e\} \in \pi$. Then, by IR-in-neighbor stability $|\pi(c) \cap V| \geqslant k$, and by individual rationality $\pi(c) \cap V$ is a clique in G. □

A.2 Appendix: Chapter 3

A.2.1 Proof of Theorem 3.1

Proof. The algorithm is similar to the one for hedonic games. We first give an informal description of our algorithm (Algorithm 7), followed by pseudocode. Again, if the input graph (N, L) is a forest, we can process each of its connected components separately, so we assume that (N, L) is a tree. We choose an arbitrary node r as the root and construct a rooted tree (N, T) by orienting the edges in L towards the leaves.

The algorithm has two different phases: the bottom-up and top-down phases.

- *Bottom-up phase*: In the bottom-up phase, we will determine *guaranteed* activity $a(i)$ and coalition $S(i)$ for every subroot i. To this end, we start with the assignment obtained by combining the previously constructed assignments $a(j)$ for the children j of i and assigning i to the void activity. We then let player i join the most preferred activity among those to which she has an IS deviation. After that we keep adding players to i's coalition $S(i)$ as long as the resulting coalition remains feasible, the player being added is willing to move, and such a deviation is acceptable for all members of i's coalition.

- *Top-down phase*: In the top-down phase, the algorithm builds a feasible assignment π, by iteratively choosing a root r' of the remaining rooted trees and reassigning the activity $a(r')$ to its coalition $S(r')$. Since each activity is copyalbe, we can always find an activity that is equivalent to $a(r')$ and has not been used by their predecessors.

We will now argue that Algorithm 7 correctly finds an individually stable feasible assignment. We start by observing the following lemma.

Lemma A.1. *For all $i \in N$, the following statements hold:*

(i) *i has no incentive to deviate to an alternative of size 1, i.e., $(a(i), |S(i)|) \succeq_i (b, 1)$ for all $b \in A$,*

(ii) *i has no IS-deviation to her children's coalitions, i.e., $(a(i), |S(i)|) \succeq_i (a(j), |S(j)| + 1)$ for any $j \in C^*(i)$, and*

(iii) *all players in $S(i)$ weakly prefer $(a(i), |S(i)|)$ to their guaranteed alternative $(a(j), |S(j)|)$.*

Proof. The statements (i) and (ii) in Lemma A.1 immediately follow from the choice of $a(i)$ in lines 4 – 11 and the stopping criterion of the **while** loop in line 14.

Algorithm 7: Finding individually stable assignments

input : tree (N, L), activity set $A = A^* \cup \{a_\emptyset\}$, $r \in N$, preference $\succeq_i$, $i \in N$
output : $\pi : N \to A$

`// Bottom-up phase: assign activities to players in a bottom-up manner.;`
make a rooted tree (N, T) with root r by orienting all the edges in L;
initialize $S(i) \leftarrow \{i\}$ and $a(i) \leftarrow a_\emptyset$ for each $i \in N$;
foreach $t = 0, \ldots,$ height(r) **do**
  foreach $i \in N$ *with* height$(i) = t$ **do**
    `// Let` i `join the favourite activity to which` i `has an IS-deviation.` set $C^*(i) = \{\, j \in \text{ch}(i) \mid (a(j), |S(j)| + 1) \succeq_k (a(j), |S(j)|)$ for all $k \in S(j) \,\}$;
    if *there exists* $b \in A$ *such that* $(b, 1) \succ_i (a(j), |S(j)| + 1)$ *for all* $j \in C^*(i)$ **then**
      find $b^* \in A$ such that $(b^*, 1) \succeq_i (b, 1)$ for all $b \in A$;
      set $S(i) \leftarrow \{i\}$ and $a(i) \leftarrow b^*$;
      else
        find $j^* \in C^*(i)$ such that $(a(j^*), |S(j^*)| + 1) \succeq_i (a(j), |S(j)| + 1)$ for all $j \in C^*(i)$;
        set $S(i) \leftarrow S(j^*) \cup \{i\}$ and $a(i) \leftarrow a(j^*)$;
    `// Add a player to` $S(i)$ `as long as the deviation is IS-feasible.;`
    while $(a(i), |S(i)| + 1) \succeq_k (a(i), |S(i)|)$ *for all* $k \in S(i)$ *and there exists* $j \in N \setminus S(i)$ *such that* j*'s parent belongs to* $S(i)$ *and* $(a(i), |S(i)| + 1) \succ_j (a(j), |S(j)|)$ **do**
      $S(i) \leftarrow S(i) \cup \{j\}$;
`// Top-down phase: relabel players with their predecessor's activities;`
set $N' \leftarrow N$ and $A' \leftarrow A^*$;
while $N' \neq \emptyset$ **do**
  choose a root r' of some connected component of the digraph $(N', T|N')$ and find an activity $b \in A' \cup \{a_\emptyset\}$ that is equivalent to $a(r')$;
  set $\pi(i) \leftarrow b$ for all $i \in S(r')$;
  set $N' \leftarrow N' \setminus S(r')$ and $A' \leftarrow A' \setminus \{b\}$;

We will now prove (iii). Assume that there is a pair of players not satisfying the condition (iii). Among such pairs, take $i \in N$ and $j \in S(i)$ with $|\text{height}(i) - \text{height}(j)|$ being the minimum. First, suppose that j joins the coalition $S(i)$ when $S(i)$ is initialized in line 12. Then, $(a(j^*), |S(j^*)|) \succeq_j (a(j), |S(j)|)$ by the minimality but we have $(a(i), |S(i)|) \succeq_j (a(j^*), |S(j^*)|)$ by the fact that adding players to $S(i)$ does not decrease j's utility, which leads to $(a(i), |S(i)|) \succeq_j (a(j), |S(j)|)$, a contradiction. Second, suppose that the player j has been added to $S(i)$ in the **while** loop in line 14. This would mean that j strictly prefers $(a(i), |S(i)|)$ to $(a(j), |S(j)|)$ at the time of termination, a contradiction. □

Now, by (iii) in Lemma A.1, it can be easily verified that $(\pi(i), |\pi_i|) \succeq_i (a(i), |S(i)|)$ for all $i \in N$. Combining this with (i), we know that at the assignment π, all players weakly prefer their alternatives to engaging alone in unused activities or the void activity. It thus remains to show that no player has an IS deviation to used activity. Suppose towards a contradiction that there is a player i who has an IS feasible deviation to the activity $\pi(j)$ of her neighbour j. If the player j is a child of i, this would mean that all players in $S(j)$ accept i, i.e., $j \in C^*(i)$, and i strictly prefers $(\pi(j), |\pi_j| + 1)$ to her alternative, namely,

$$(a(j), |S(j)| + 1) = (\pi(j), |\pi_j| + 1) \succ_i (\pi(i), |\pi_i|) \succeq_i (a(i), |S(i)|),$$

contradicting (ii) in Lemma A.1. If the player j is the parent of i in the rooted tree T, j must have been added to π_j in the **while** loop in line 14, a contradiction. We conclude that no player has an IS-deviation to used activities, and hence π is individually stable.

It remains to analyze the running time of Algorithm 7. Consider the execution of the algorithm for a fixed player i. Let $c = |\text{ch}(i)|$ and $d = |\text{desc}(i)|$. Line 4 requires at most d queries: no descendant of i is queried more than once. Lines 5 – 10 require p queries. Moreover, at each iteration of the **while** loop in lines 12–14 at least one player joins $S(i)$, so there are at most d iterations, in each iteration we consider at most d candidates, and for each candidate we perform at most d queries. Summing over all players, we conclude that the number of queries for the bottom-up phase is bounded by $O(n(n^2 + p))$. It is immediate that the top-down phase can be done in polynomial time. This completes the proof of the theorem. □

A.2.2 Proof of Theorem 3.7

Proof. We give separate proofs for the three types of networks considered: paths, stars, and graphs with small components.

Paths. We prove hardness via a reduction from PATH RAINBOW MATCHING.

Construction. Given an instance $(G, \mathcal{C}, \phi, k)$ of PATH RAINBOW MATCHING where $|\mathcal{C}| = q$, we create a vertex player v for each $v \in V$ and an edge player e for each $e \in E$. To create the social network, we first construct the graph (N_G, L_G) as defined in the proof for Theorem 3.4. To the right of the graph (N_G, L_G), we attach a path consisting of garbage collectors $\{g_1, g_2, \ldots, g_{q-k}\}$ and q copies (N_c, L_c) of the empty-IS instance of Example 3.3 where $N_c = \{c_1, c_2, c_3\}$ and $L_c = \{\{c_1, c_2\}, \{c_2, c_3\}\}$ for each $c \in \mathcal{C}$. For each color $c \in \mathcal{C}$, we introduce a color activity c, and introduce additional activities $a(c)$ and $b(c)$. Each vertex player v approves color activities $\phi(e)$ of its adjacent edges e with size 3; each edge player e approves the color activity $\phi(e)$ of its color with size 3; each garbage collector g_i approves any color activity c with size 1; finally, the preference for players in N_c ($c \in \mathcal{C}$) is cyclic and given by

$$c_1 : (b(c), 2) \succ (a(c), 1) \succ (c, 3) \succ (c, 2) \succ (c, 1) \succ (a_\emptyset, 1)$$
$$c_2 : (c, 3) \succ (c, 2) \succ (a(c), 2) \succ (b(c), 2) \succ (b(c), 1) \succ (a_\emptyset, 1)$$
$$c_3 : (c, 3) \succ (a(c), 2) \succ (a(c), 1) \succ (a_\emptyset, 1)$$

Correctness. We will now prove that the following three statements are equivalent.

(i) There exists an individually stable feasible assignment.

(ii) There exists a core stable feasible assignment.

(iii) G contains a rainbow matching of size at least k.

(i) or (ii) $\implies$ (iii): We will show that if one of two stable solutions exists, then there exists a rainbow matching of size at least k. First, suppose that there is a core or individually stable feasible assignment $\pi : N \to A$. Let $M = \{\, e \in E \mid \pi(e) \in \mathcal{C} \,\}$. We will show that M is a rainbow matching of size at least k. To see this, notice that at π, all the color activities should be played outside N_c's, since otherwise no core or individually stable assignment would exist as we have seen in Example 3.3. Further, at most $q - k$ colour activities are played among the garbage collectors, which means that at least k colour activities should be assigned to vertex and edge players. The only individual rational way to do this is to select triples of the form (u, e, v) where $e = \{u, v\} \in E$ and assign to them their colour activity $\phi(e)$; thus, M is a rainbow matching of size at least k.
(iii) $\implies$ (i) and (ii): Suppose that there exists a rainbow matching M of size k. We construct a feasible assignment π where for each $e = \{u, v\} \in M$ we set $\pi(e) = \pi(u) = \pi(v) = \phi(e)$, each garbage collector g_i, $i \in [q - k]$, is arbitrarily assigned to one of the remaining $q - k$

color activities, each pair of c_2 and c_3 ($c \in \mathcal{C}$) is assigned to $a(c)$, and the remaining players are assigned to the void activity. The assignment π is individually stable, since every garbage collector as well as every edge or vertex player assigned to a color activity is allocated their top alternative, and no remaining player has an IS feasible deviation. Similarly, it can be easily verified that π is core stable.

Stars. Again, we reduce from a restricted variant of MMM where the graph is a bipartite graph. We have seen that computing a core stable outcome is NP-hard for stars in Theorem 3.12; hence, we only provide a hardness proof for individual stability.

Construction. Given a bipartite graph (U, V, E) and an integer k, we construct an instance of gGASP on a path as follows. We create a star with center c and the $|V| + 2$ leaves: one leaf for each vertex $v \in V$ plus two other players s_1 and s_2. We then introduce an activity u for each $u \in U$, and four other simple activities a, x, y, and z.

A player $v \in V$ approves $(u, 1)$ for each $u \in U$ such that $\{u, v\} \in E$ as well as $(a, |V| - k + 1)$ and prefers the former to the latter. That is, $(u, 1) \succ_v (a, |V| - k + 1)$ for any $u \in U$ with $\{u, v\} \in E$; v is indifferent among activities associated with its neighbors in the graph, that is, $(u, 1) \sim_v (u', 1)$ for all $u, u' \in U$ such that $\{u, v\}, \{u', v\} \in E$. The center player c strictly prefers $(a, |V| - k + 1)$ to any other alternative, and has the same cyclic preferences over the alternatives of x, y, and z as in Example 3.3 together with players s_1 and s_2; explicitly, the preferences are given by

$$
\begin{aligned}
&s_1 : (y, 2) \succ (x, 1) \succ (z, 3) \succ (z, 2) \succ (z, 1) \succ (a_\emptyset, 1) \\
&c : (a, |V| - k + 1) \succ (z, 3) \succ (z, 2) \succ (x, 2) \succ (y, 2) \succ (y, 1) \succ (a_\emptyset, 1) \\
&s_2 : (z, 3) \succ (x, 2) \succ (x, 1) \succ (a_\emptyset, 1).
\end{aligned}
$$

Here, s_1's (respectively, the center c and the player s_2) preference corresponds to the one for player 1 (respectively, player 2 and player 3) in Example 3.3.

Correctness. We will show that there exists an individually stable feasible assignment if and only if G contains a maximal matching of size at most k.

Suppose that there exists an individually stable feasible assignment π and let $M = \{ \{\pi(v), v\} \mid v \in V \wedge \pi(v) \in U \}$. We will show that M is a maximal matching of size at most k. By stability, the center player and $|V| - k$ vertex players are assigned to the activity a; otherwise, no individually stable outcome would exist as we have seen in Example 3.3; thus, $|M| \leq k$. Notice further that M is a matching since each vertex player v plays at most one activity, and by individual rationality each vertex activity should be assigned to at most one player. Now suppose towards a contradiction that M is not maximal, i.e., there exists an edge $\{u, v\} \in E$ such that $u \in U$, $v \in V$, and $M \cup \{u, v\}$ is a matching. This would mean

that π assigns no player to activity u and no vertex activity to player v. Hence, the player v has an IS-deviation to the vertex activity u, contradicting the stability of π.

Conversely, suppose that G admits a maximal matching M with at most k edges. We construct a feasible assignment π by setting $\pi(v) = u$ for each $\{u, v\} \in M$, and assigning $|V| - k$ non-matched vertex players and the center to a, assigning s_1 to x, and assigning the remaining players to the void activity. The center c is allocated to her top alternative; hence no player has an IS-deviation to an activity a and the center does not want to deviate to a used vertex activity activity. No vertex player v has an IS-deviation to an unused vertex activity u, since if such a pair $\{u, v\}$ existed, this would mean that $\{u, v\}$ is not included in M, and hence $M \cup \{u, v\}$ forms a matching, which contradicts the maximality of M. Hence, π is individually stable. This completes the proof.

Small components. We reduce from (3,B2)-SAT.

Construction. Consider a formula ϕ with variable set X and clause set C, where for each variable $x \in X$ we write x_1 and x_2 for the two positive occurrences of x, and $\bar{x}_1$ and $\bar{x}_2$ for the two negative occurrences of x. We construct an instance of gGASP as follows. For each variable $x \in X$, we introduce three variable players $v_1(x)$, $v_2(x)$, and $v_3(x)$; and we also introduce three positive variable players $p_1(x)$, $p_2(x)$, and $p_3(x)$, and three negative variable players $\bar{p}_1(x)$, $\bar{p}_2(x)$, and $\bar{p}_3(x)$. For each clause $c \in C$, we introduce three players $N_c = \{c_1, c_2, c_3\}$. The network consists of one component for each clause $c \in C$: a star with center c_2 and leaves c_1 and c_3, and of three components for each variable $x \in X$:

- a star with center $v_2(x)$ and leaves $v_1(x)$ and $v_3(x)$,
- a star with center $p_2(x)$ and leaves $p_1(x)$ and $p_3(x)$, and
- a star with center $\bar{p}_2(x)$ and leaves $\bar{p}_1(x)$ and $\bar{p}_3(x)$.

Hence, the size of each connected component of this graph is at most 3. Corresponding to the variable occurrences, we introduce two positive literal activities x_1 and x_2, two negative literal activities $\bar{x}_1$ and $\bar{x}_2$ for each $x \in X$; we also introduce one variable activity x, and eight further activities $y(x), z(x)$, $a(x), b(x), c(x)$, $\bar{a}(x), \bar{b}(x)$, and $\bar{c}(x)$. Thus, the set of activities given by

$$A^* = \bigcup_{x \in X} \{x_1, x_2, \bar{x}_1, \bar{x}_2, x, y(x), z(x), a(x), b(x), c(x), \bar{a}(x), \bar{b}(x), \bar{c}(x)\}.$$

For each $x \in X$, the preferences of variable players are given as follows:

$$v_1(x):\ (y(x),2) \succ (x,1) \succ (z(x),3) \succ (z(x),2) \succ (z(x),1) \succ (a_\emptyset,1),$$
$$v_2(x):\ (z(x),3) \succ (z(x),2) \succ (x,2) \succ (y(x),2) \succ (y(x),1) \succ (a_\emptyset,1),$$
$$v_3(x):\ (z(x),3) \succ (x,2) \succ (x,1) \succ (a_\emptyset,1).$$

Notice that the preferences are cyclic; thus, in a core or individually stable assignment, one of the activities $x, y(x)$, and $z(x)$ must be allocated outside of the variable players $v_1(x)$, $v_2(x)$, and $v_3(x)$. For each $x \in X$, the preferences of the three positive variable players $p_1(x)$, $p_2(x)$, and $p_3(x)$ are given as follows:

$$p_1(x):\ (x,3) \sim (x_1,1) \succ (b(x),2) \succ (a(x),1) \succ (c(x),3) \succ (c(x),2) \succ (c(x),1) \succ (a_\emptyset,1),$$
$$p_2(x):\ (x,3) \sim (x_2,1) \succ (c(x),3) \succ (c(x),2) \succ (a(x),2) \succ (b(x),2) \succ (b(x),1) \succ (a_\emptyset,1),$$
$$p_3(x):\ (x,3) \succ (c(x),3) \succ (a(x),2) \succ (a(x),1) \succ (a_\emptyset,1).$$

Similarly, for each $x \in X$, the preferences of the negative variable players $\bar{p}_1(x)$, $\bar{p}_2(x)$, and $\bar{p}_3(x)$ are given as follows:

$$\bar{p}_1(x):\ (x,3) \sim (\bar{x}_1,1) \succ (\bar{b}(x),2) \succ (\bar{a}(x),1) \succ (\bar{c}(x),3) \succ (\bar{c}(x),2) \succ (\bar{c}(x),1) \succ (a_\emptyset,1),$$
$$\bar{p}_2(x):\ (x,3) \sim (\bar{x}_2,1) \succ (\bar{c}(x),3) \succ (\bar{c}(x),2) \succ (\bar{a}(x),2) \succ (\bar{b}(x),2) \succ (\bar{b}(x),1) \succ (a_\emptyset,1),$$
$$\bar{p}_3(x):\ (x,3) \succ (\bar{c}(x),3) \succ (\bar{a}(x),2) \succ (\bar{a}(x),1) \succ (a_\emptyset,1).$$

Again, observe that the preferences of each triple contains a cyclic relation, and hence in a core stable assignment and an individually stable assignment, there are three possible cases:

- all the three players $p_1(x)$, $p_2(x)$, and $p_3(x)$ are assigned to x_1, x_2, and $a(x)$, respectively, and players $\bar{p}_1(x)$, $\bar{p}_2(x)$, and $\bar{x}$ are assigned to activities $\bar{x}_1$, $\bar{x}_2$, and $\bar{a}(x)$, respectively;
- all the three players $p_1(x)$, $p_2(x)$, and $p_3(x)$ are assigned to x, and players $\bar{p}_1(x)$, $\bar{p}_2(x)$, and $\bar{x}$ are assigned to activities $\bar{x}_1$, $\bar{x}_2$, and $\bar{a}(x)$, respectively; or,
- all the players $\bar{p}_1(x)$, $\bar{p}_2(x)$, and $\bar{p}_3(x)$ are assigned to x, and players $p_1(x)$, $p_2(x)$, and $p_3(x)$ are assigned to activities x_1, x_2, and $a(x)$, respectively.

Later we will see that the last two cases can only occur in a stable assignment.

For each clause $c \in C$ where ℓ_1^c, ℓ_2^c, and ℓ_3^c are the literals in a clause c, the preferences for clause players c_1, c_2, and c_3 are again cyclic and given as follows:

$$c_1:\ (\ell_2^c,2) \succ (\ell_1^c,1) \succ (\ell_3^c,3) \succ (\ell_3^c,2) \succ (\ell_3^c,1) \succ (a_\emptyset,1),$$
$$c_2:\ (\ell_3^c,3) \succ (\ell_3^c,2) \succ (\ell_1^c,2) \succ (\ell_2^c,2) \succ (\ell_2^c,1) \succ (a_\emptyset,1),$$
$$c_3:\ (\ell_3^c,2) \succ (\ell_1^c,3) \succ (\ell_1^c,1) \succ (a_\emptyset,1).$$

If there exists a core or individually stable outcome, it must be the case that at least one of the literal activities ℓ_1^c, ℓ_2^c, and ℓ_3^c must be used outside of the three players c_1, c_2, and c_3; otherwise, no feasible assignment would be individually stable.

Correctness. Now we will show that the following three statements are equivalent.

(i) There exists an individually stable feasible assignment.

(ii) There exists a core stable feasible assignment.

(iii) The formula ϕ is satisfied by some truth assignment.

(i) or (ii) $\implies$ (iii): Suppose that there exists a core or individually stable feasible assignment π. By stability, for each variable $x \in X$, one of the activities x, $y(x)$, and $z(x)$ must be assigned outside of the variable players $v_1(x)$, $v_2(x)$, and $v_3(x)$. The only individually rational way to do this is to assign the variable activity x to either the positive variable players $p_1(x)$, $p_2(x)$, and $p_3(x)$, or the negative variable players $\bar{p}_1(x)$, $\bar{p}_2(x)$, and $\bar{p}_3(x)$, meaning that either a pair of positive literal activities x_1 and x_2 or a pair of negative literal activities $\bar{x}_1$ and $\bar{x}_2$ should be assigned to the corresponding pair of variable players. Further, for each clause c, at least one of the literal activities ℓ_1^c, ℓ_2^c, and ℓ_3^c should be played outside of the clause players c_1, c_2, and c_3. Then, take the truth assignment that sets the variables x to *true* if their positive variable players $p_1(x)$ and $p_2(x)$ are assigned to positive literal activities x_1 and x_2; otherwise, x is set to *false*; this can be easily seen to satisfy ϕ.
(iii) $\implies$ (i) and (ii): Suppose that there exists a truth assignment that satisfies ϕ. We construct a feasible assignment π as follows. First, for every variable $x \in X$, we assign $z(x)$ to the variable players $v_1(x)$, $v_2(x)$, and $v_3(x)$. Then, for each variable x that is set to *true*, we assign positive literal activities x_1, x_2, and $a(x)$ to positive variable players $p_1(x)$, $p_2(x)$, and $p_3(x)$, respectively, and assign a variable activity x to players $\bar{p}_1(x)$, $\bar{p}_2(x)$, and $\bar{p}_3(x)$. For each variable x that is set to *false*, we assign negative literal activities $\bar{x}_1$, $\bar{x}_2$, and $\bar{a}(x)$ to negative literal players $\bar{p}_1(x)$, $\bar{p}_2(x)$, and $\bar{p}_3(x)$, respectively, and assign a variable activity x to players $p_1(x)$, $p_2(x)$, and $p_3(x)$. Note that this procedure uses at least one of the literal activities ℓ_1^c, ℓ_2^c and ℓ_3^c of each clause c, since the given truth assignment satisfies ϕ. Then, for each clause $c \in C$, we assign activities to the clause players c_1, c_2, and c_3 as follows.

- If all the activities are already assigned in a previous step, then we assign the void activity to all the clause players c_1, c_2, and c_3.
- If ℓ_1^c is already assigned and ℓ_3^c is not assigned, then we assign ℓ_3^c to all the clause players c_1, c_2, and c_3.

- If ℓ_3^c is already assigned and ℓ_1^c is not assigned, then we assign ℓ_1^c to the players c_2, and c_3.
- If both ℓ_1^c and ℓ_3^c are already assigned and ℓ_2^c is not assigned, then we assign ℓ_2^c to the player c_1 and c_2.

The resulting assignment π of players to activities is individually stable, because no variable player wishes to change their alternative and no player of each N_c has an IS feasible deviation. Similarly, it can be easily verified that π is core stable. □

A.2.3 Proof of Theorem 3.9

Proof. Again, we give a dynamic programming algorithm. Suppose our graph (N, L) has k connected components $(N_1, L_1), (N_2, L_2), \ldots, (N_k, L_k)$. For each $i \in [k]$, each set $B \subseteq A^*$ of activities assigned to N, and each set $B' \subseteq B$ of activities assigned to $\bigcup_{j=1}^{i} N_j$, we let $\mathsf{CR}[i, B, B']$ to be *true* if there exists an individually rational feasible assignment $\pi : \bigcup_{j=1}^{i} N_j \to A$ such that

- the set of activities assigned to $\bigcup_{j=1}^{i} N_j$ is exactly B'; and
- no connected subset $S \subseteq \bigcup_{j=1}^{i} N_j$ together with an activity in $B' \cup (A^* \setminus B)$ strongly blocks π

Otherwise, $\mathsf{CR}[i, B, B']$ is *false*.

For $i = 1$, each $B \subseteq A^*$, and each $B' \subseteq B$, we compute the value of $\mathsf{CR}[1, B, B']$ by trying all possible mappings $\pi : N \to B' \cup \{a_\emptyset\}$, and checking whether it is an individually rational feasible assignment using all activities in B' and such that none of the connected subsets $S \subseteq N_1$ together with an activity in $B' \cup (A^* \setminus B)$ strongly blocks π.

For $i = 2, 3, \ldots, k$, each $B \subseteq A^*$, and $B' \subseteq B$, we set $\mathsf{CR}[i, B, B']$ to *true* if there exists a bipartition of B' into P and Q such that $\mathsf{CR}[i-1, B, P]$ is *true* and there exists an individually rational feasible assignment $\pi : N_i \to Q \cup \{a_\emptyset\}$ such that each activity in Q is assigned to some player in N_i, and none of the connected subsets $S \subseteq N_i$ together with an activity in $Q \cup (A^* \setminus B)$ strongly blocks π.

It is not difficult to see that a core stable solution exists if and only if $\mathsf{CR}[k, B, B]$ is *true* for some $B \subseteq A^*$. If this is the case, such a stable feasible assignment can be found using standard dynamic programming techniques. The size of the dynamic programming table is at most $4^p n$. There are at most 2^p bipartitions of B', and for each bipartition, we consider $O(p^c)$ possible assignments; further, for each assignment, we have shown that there is an $O(pc^3)$ time algorithm to check the existence of a strongly blocking coalition. Thus, each entry can be filled in time $O(2^p p^{c+1} c^3)$. This completes the proof. □

A.2.4 Proof of Theorem 3.11

Proof. Again, given a tree (N, L), we choose an arbitrary node as the root and construct a rooted tree by orienting the edges in L towards the leaves; we denote by $\text{ch}(i)$ the set of children of i and by $\text{desc}(i)$ the set of descendants of i (including i).

We process the nodes from the leaves to the root. For each $i \in N$, each $B \subseteq A^*$, each $B' \subseteq B$, each $(a, k) \in B' \times [n] \cup \{(a_\emptyset, 1)\}$, and each $t \in [k]$, we will again check whether there is a partial assignment of B' to i's descendants that is extensible to a stable assignment, assuming that the whole players are assigned to the activities in B. Here, there are three cases: first, the i's coalition may admit a deviation from their descendants but some player who does not want to increase the size of the coalition may join the coalition later on (*possibly stable*); second, there is already a player in the i's coalition who does not want to increase the size of the coalition (*definitely stable*); third, no descendant of i's coalition wants to deviate to it (*weakly stable*). For all three concepts, all coalitions of descendants of i that do not involve i are immune to any IS-deviation.

Formally, we first let $\mathsf{PS}[i, B, B', (a, k), t]$ be *true* if there exists a feasible assignment $\pi : N \to A$ where

(i) the set of activities assigned to players in $\text{desc}(i)$ is exactly B';

(ii) player i is assigned to a and is in a coalition with k other players;

(iii) exactly t players in $\text{desc}(i)$ belong to the same group as i;

(iv) the t players in $\text{desc}(i) \cap \pi_i$ weakly prefer (a, k) to $(b, 1)$ for each $b \in A \setminus B$, and has no IS-deviation to the activities assigned to players in $D_i \setminus \pi_i$; and

(v) the players in $\text{desc}(i) \setminus \pi_i$ weakly prefer their alternative under π to engaging alone in any of the activities in $A \setminus B$, and have no IS-deviations to activities in $B' \setminus \{a\}$.

Otherwise, we let $\mathsf{PS}[i, B, B', (a, k), t]$ be *false*. Second, we let $\mathsf{DS}[i, B, B', (a, k), t]$ be *true* if there exists a feasible assignment $\pi : N \to \mathbb{R}$ such that (i) - (v) hold, and (vi) $a = a_\emptyset$ or some player in $\text{desc}(i) \cap \pi_i$ strictly prefers (a, k) to $(a, k + 1)$; we let $\mathsf{DS}[i, B, B', (a, k), t]$ be *false* otherwise. Third, we let $\mathsf{WS}[i, B, B', (a, k), t]$ be *true* if there exists a feasible assignment $\pi : N \to \mathbb{R}$ such that (i) - (v) hold, and (vii) $a = a_\emptyset$ or no player in $\text{desc}(i) \setminus \pi_i$ strictly prefers $(a, k + 1)$ to their alternative under π; we let $\mathsf{WS}[i, B, B', (a, k), t]$ be *false* otherwise.

By construction, our instance admits an individually stable assignment if and only if there exists a combination of the arguments B, B', and (a, k) such that $\mathsf{DS}[t, B, B', (a, k), k]$

is *true* or $\mathsf{WS}[r, B, B', (a, k), k]$ is *true*, where r is the root. In what follows, we will describe how to compute each entry of all the three tables in a bottom-up manner.

Leaves. If i is a leaf, we set both $\mathsf{PS}[i, B, B', (a, k), t]$ and $\mathsf{WS}[i, B, B', (a, k), t]$ to *true* if $B' = \{a\}$, $t = 1$, and i weakly prefers (a, k) to any other alternative $(b, 1)$ such that $b \in A \setminus B$; otherwise, we set both $\mathsf{PS}[i, B, B', (a, k), t]$ and $\mathsf{WS}[i, B, B', (a, k), t]$ to *false*. We set $\mathsf{DS}[i, B, B', (a, k), t]$ to *true* if $\mathsf{PS}[i, B, B', (a, k), t]$ is true, and $a = a_\emptyset$ or i strictly prefers (a, k) to $(a, k + 1)$; otherwise, we set $\mathsf{DS}[i, B, B', (a, k), t]$ to *false*.

Internal vertices. Now consider the case where i is an internal vertex; we assume that all the values for i's descendants have been computed. We first check whether i strictly prefers some alternative $(b, 1)$ such that $b \in A \setminus B$ to (a, k); if so, we set $\mathsf{PS}[i, B, B', (a, k), t]$, $\mathsf{DS}[i, B, B', (a, k), t]$, and $\mathsf{WS}[i, B, B', (a, k), t]$ to *false*. Otherwise, we proceed and check for each partition $\mathcal{P}$ of $B' \setminus \{a\}$ and each of its permutations whether there is an allocation of each activity set $P \in \mathcal{P}$ to some subtree rooted at i's child that gives rise to an assignment with the conditions described before. One can show that determining the existence of a desired assignment can be computed in polynomial time. Then, similarly to the proof of Theorem 3.11, our problem for trees is in FPT and, moreover, we can easily extend this to arbitrary forests.

Now let us fix a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$ of $B' \setminus \{a\}$. Again, without loss of generality, we consider an ordering $P_1, P_2, \ldots, P_{|\mathcal{P}|}$ and seek to assign each activity set to the subtrees in that order. It remains to show that there exists a polynomial time algorithm that determines whether there exists a feasible assignment $\pi : N \to A$ such that the conditions (i) - (v) (respectively, (i) - (vi), and (i) - (v), (vii)) hold, and each activity set in $\mathcal{P}$ is assigned to the players in $\mathrm{desc}(j)$ for some $j \in \mathrm{ch}(i)$ in such a way that for each $q \in [|\mathcal{P}|]$ and each $c \in [|\mathrm{ch}(i)|]$, P_q is assigned to $\mathrm{desc}(j_c)$ only if the prefixes $P_1, P_2, \ldots, P_{q-1}$ are assigned to the subtrees $\mathrm{desc}(j_1), \mathrm{desc}(j_2), \ldots, \mathrm{desc}(j_{c-1})$. To this end, we give a dynamic programming for a respective problem; we denote by P_0 the empty activity set.

Computation of $\mathsf{PS}[i, B, B', (a, k), t]$: First, for each $c \in [|\mathrm{ch}(i)|]$ and $q \in [0, |\mathcal{P}|]$, we will check whether the activity sets $P_0, P_1, \ldots, P_q$ can be assigned to the subtrees rooted at $j_1, j_2, \ldots, j_c$, and exactly ℓ players can be assigned to the activity a; we refer to this subproblem by $TP[j_c, P_q, \ell]$. We initialize $TP[j_1, P_q, \ell]$ to *true* if one of the following statements holds:

- the empty activity set P_0 can be allocated to the first subtree $\mathrm{desc}(j_1)$, i.e., $q = 0$, $\ell = 0$, and $\mathsf{PS}[j_1, B, \emptyset, (a_\emptyset, 1), 1]$ is *true*;
- only the activity a can be assigned to ℓ players in $\mathrm{desc}(j_1)$, i.e., $q = 0$, $\ell \geq 1$, and $\mathsf{PS}[j_1, B, \{a\}, (a, k), \ell]$ is *true*;

- only the activity set P_1 can be assigned to players in $\text{desc}(j_1)$, i.e., $q = 1$, $\ell = 0$, and there exists an alternative $(b, x) \in P_1 \times [n] \cup \{(a_\emptyset, 1)\}$ such that $\mathsf{DS}[j_1, B, P_1, (b, x), x]$ is *true*, or $\mathsf{WS}[j_1, B, P_1, (b, x), x]$ is *true* and ($b = a_\emptyset$ or i weakly prefers (a, k) to $(b, x + 1)$);

- P_1 can be assigned to players in $\text{desc}(j_1)$ while activity a can be assigned to ℓ players from $\text{desc}(j_1)$, i.e., $q = 1$, $\ell \geq 1$, and $\mathsf{PS}[j_1, B, P_1 \cup \{a\}, (a, k), \ell]$ is *true*.

We set $TP[j_1, P_q, \ell]$ to *false* otherwise. Then, we iterate through $j_1, j_2, \ldots, j_{|\text{ch}(i)|}$ and $P_0, P_1, \ldots, P_{|\mathcal{P}|}$, and update $T[j_c, P_q, \ell]$: for each $c \in [|\text{ch}(i)|]$, for each $q \in [|\mathcal{P}|]$, and for each $\ell \in [0, t]$, we set $T[j_c, P_q, \ell]$ to *true* if one of the following statements holds:

- $P_1, P_2, \ldots, P_q$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ with ℓ players from the subtrees being assigned to the activity a, and the void activity can be assigned to the subtree $\text{desc}(j_c)$, i.e., both $TP[j_{c-1}, P_q, \ell]$ and $\mathsf{PS}[j_c, B, \emptyset, (a_\emptyset, 1), 1]$ are *true*;

- $P_1, P_2, \ldots, P_q$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ while the activity a is assigned to x players from the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ and to $\ell - x$ players from the subtree $\text{desc}(j_c)$, i.e., $\ell \geq 2$, and there exists an $x \in [\ell - 1]$ such that both $TP[j_{c-1}, P_q, \ell - x]$ and $\mathsf{PS}[j_c, B, \{a\}, (a, k), x]$ are *true*;

- $P_1, P_2, \ldots, P_{q-1}$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ with ℓ players from the subtrees being assigned to the activity a, and only the activity set P_q can be assigned to the subtree $\text{desc}(j_c)$, i.e., $TP[j_{c-1}, P_{q-1}, \ell]$ is *true*, and there exists an alternative $(b, x) \in P_q \times [n] \cup \{(a_\emptyset, 1)\}$ such that $\mathsf{DS}[j_c, B, P_q, (b, x), x]$ is *true*, or $\mathsf{WS}[j_c, B, P_q, (b, x), x]$ is *true* and ($b = a_\emptyset$ or i weakly prefers (a, k) to $(b, x + 1)$);

- $P_1, P_2, \ldots, P_{q-1}$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$, and the activity set P_q can be assigned to the subtree $\text{desc}(j_c)$ while the activity a is assigned to x players from the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ and to $\ell - x$ players from the subtree $\text{desc}(j_c)$, i.e., $\ell \geq 2$, and there exists an $x \in [\ell - 1]$ such that both $TP[j_{c-1}, P_{q-1}, \ell - x]$ and $\mathsf{PS}[j_c, B, P_q \cup \{a\}, (a, k), x]$ are *true*.

We set $TP[j_c, P_q, \ell]$ to *false* otherwise.

Computation of $\mathsf{DS}[i, B, B', (a, k), t]$: Second, for each $c \in [|\text{ch}(i)|]$ and $q \in [0, |\mathcal{P}|]$, we will check whether the activity sets $P_0, P_1, \ldots, P_q$ can be assigned to the subtrees rooted

at $j_1, j_2, \ldots, j_c$, and exactly ℓ players can be assigned to the activity a, and ($a = a_\emptyset$ or some of the ℓ players strictly prefers (a, k) to $(a, k+1)$); we refer to this by $TD[j_c, P_q, \ell]$.

If i strictly prefers (a, k) to $(a, k{+}1)$, then we set each value $TD[j_c, P_q, \ell]$ to $TP[j_1, P_q, \ell]$. Otherwise, we compute $TD[j_c, P_q, \ell]$ as follows. We initialize $TD[j_1, P_q, \ell]$ to *true* if one of the following statements holds:

- only the activity a can be assigned to ℓ players in desc(j_1), i.e., $q = 0$, $\ell \geq 1$, and $\mathsf{DS}[j_1, B, \{a\}, (a, k), \ell]$ is *true*;
- P_1 can be assigned to players in desc(j_1) while activity a can be assigned to ℓ players from desc(j_1), i.e., $q = 1$, $\ell \geq 1$, and $\mathsf{DS}[j_1, B, P_1 \cup \{a\}, (a, k), \ell]$ or) is *true*.

We set $TD[j_1, P_q, \ell]$ to *false* otherwise. Then, for each $c \in [|\text{ch}(i)|]$, for each $q \in [|\mathcal{P}|]$, and for each $\ell \in [0, t]$, we set $TD[j_c, P_q, \ell]$ to *true* if one of the following statements holds:

- $P_1, P_2, \ldots, P_q$ can be assigned to the subtrees desc(j_1), desc(j_2), $\ldots$, desc(j_{c-1}) with ℓ players from the subtrees being assigned to the activity a, and the void activity can be assigned to the subtree desc(j_c), i.e., both $TD[j_{c-1}, P_q, \ell]$ and $\mathsf{PS}[j_c, B, \emptyset, (a_\emptyset, 1), 1]$ are *true*;
- $P_1, P_2, \ldots, P_q$ can be assigned to the subtrees desc(j_1), desc(j_2), $\ldots$, desc(j_{c-1}) while the activity a is assigned to x players from the subtrees desc(j_1), desc(j_2), $\ldots$, desc(j_{c-1}) and to $\ell - x$ players from the subtree desc(j_c), i.e., $\ell \geq 2$, and there exists an $x \in [\ell - 1]$ such that both $TP[j_{c-1}, P_q, \ell - x]$ and $\mathsf{DS}[j_c, B, \{a\}, (a, k), x]$ are *true*, or both $TD[j_{c-1}, P_q, \ell - x]$ and $\mathsf{PS}[j_c, B, \{a\}, (a, k), x]$ are *true*;
- $P_1, P_2, \ldots, P_{q-1}$ can be assigned to the subtrees desc(j_1), desc(j_2), $\ldots$, desc(j_{c-1}) with ℓ players from the subtrees being assigned to the activity a, and only the activity set P_q can be assigned to the subtree desc(j_c), i.e., $TD[j_{c-1}, P_{q-1}, \ell]$ is *true*, and there exists an alternative $(b, x) \in P_q \times [n] \cup \{(a_\emptyset, 1)\}$ such that $\mathsf{DS}[j_c, B, P_q, (b, x), x]$ is *true*, or $\mathsf{WS}[j_c, B, P_q, (b, x), x]$ is *true* and ($b = a_\emptyset$ or i weakly prefers (a, k) to $(b, x + 1)$);
- $P_1, P_2, \ldots, P_{q-1}$ can be assigned to the subtrees desc(j_1), desc(j_2), $\ldots$, desc(j_{c-1}) and P_q can be assigned to the subtree desc(j_c) while the activity a is assigned to x players from the subtrees desc(j_1), desc(j_2), $\ldots$, desc(j_{c-1}) and to $\ell - x$ players from the subtree desc(j_c), i.e., $\ell \geq 2$, and there exists an $x \in [\ell - 1]$ such that both $TD[j_{c-1}, P_{q-1}, \ell - x]$ and $\mathsf{PS}[j_c, B, P_q \cup \{a\}, (a, k), x]$ are *true*, or both $TP[j_{c-1}, P_{q-1}, \ell - x]$ and $\mathsf{DS}[j_c, B, P_q \cup \{a\}, (a, k), x]$ are *true*.

We set $TD[j_c, P_q, \ell]$ to *false* otherwise.

Computation of $\mathsf{WS}[i, B, B', (a, k), t]$: Third, for each $c \in [|\text{ch}(i)|]$ and $q \in [0, |\mathcal{P}|]$, we will check whether the activity sets $P_0, P_1, \ldots, P_q$ can be assigned to the subtrees rooted at $j_1, j_2, \ldots, j_c$, and exactly ℓ players can be assigned, and no player in $\text{desc}(j)$ has an incentive to deviate to i's coalition; we refer to this subproblem by $TW[j_c, P_q, \ell]$. We initialize $TW[j_1, P_q, \ell]$ to *true* if one of the following statements holds:

- the empty activity set P_0 can be allocated to the first subtree $\text{desc}(j_1)$, i.e., $q = 0$, $\ell = 0$, $\mathsf{PS}[j_1, B, \emptyset, (a_\emptyset, 1), 1]$ is *true*, and ($a = a_\emptyset$ or j_1 weakly prefers $(a_\emptyset, 1)$ to $(a, k+1)$);

- only the activity a can be assigned to ℓ players in $\text{desc}(j_1)$, i.e., $q = 0$, $\ell \geq 1$, and $\mathsf{WS}[j_1, B, \{a\}, (a, k), \ell]$ is *true*;

- only the activity set P_1 can be assigned to players in $\text{desc}(j_1)$, i.e., $q = 1$, $\ell = 0$, and there exists an alternative $(b, x) \in P_1 \times [n] \cup \{(a_\emptyset, 1)\}$ such that (1). $\mathsf{DS}[j_1, B, P_1, (b, x), x]$ is *true*, or ($\mathsf{WS}[j_1, B, P_1, (b, x), x]$ is *true* and ($b = a_\emptyset$ or i weakly prefers (a, k) to $(b, x+1)$)), and (2). ($a = a_\emptyset$ or j_1 weakly prefers (b, x) to $(a, k+1)$);

- P_1 can be assigned to players in $\text{desc}(j_1)$ while activity a can be assigned to ℓ players from $\text{desc}(j_1)$, i.e., $q = 1$, $\ell \geq 1$, and $\mathsf{WS}[j_1, B, P_1 \cup \{a\}, (a, k), \ell]$ is *true*.

We set $TW[j_1, P_q, \ell]$ to *false* otherwise. Then, for each $c \in [|\text{ch}(i)|]$, for each $q \in [|\mathcal{P}|]$, and for each $\ell \in [0, t]$, we set $TW[j_c, P_q, \ell]$ to *true* if one of the following statements holds:

- $P_1, P_2, \ldots, P_q$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ with ℓ players from the subtrees being assigned to the activity a, and the void activity can be assigned to the subtree $\text{desc}(j_c)$, i.e., both $TW[j_{c-1}, P_q, \ell]$ and $\mathsf{PS}[j_c, B, \emptyset, (a_\emptyset, 1), 1]$ are *true*, and $a = a_\emptyset$ or player j_c weakly prefers $(a_\emptyset, 1)$ to $(a, k+1)$;

- $P_1, P_2, \ldots, P_q$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ while the activity a is assigned to x players from the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ and to $\ell - x$ players from the subtree $\text{desc}(j_c)$, i.e., $\ell \geq 2$, and there exists an $x \in [\ell - 1]$ such that both $TW[j_{c-1}, P_q, \ell - x]$ and $\mathsf{WS}[j_c, B, \{a\}, (a, k), x]$ are *true*;

- $P_1, P_2, \ldots, P_{q-1}$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ with ℓ players from the subtrees being assigned to the activity a, and P_q can be assigned to the subtree $\text{desc}(j_c)$ with no player from the subtree being assigned to a, i.e., $TW[j_{c-1}, P_{q-1}, \ell]$ is *true*, and there exists an alternative $(b, x) \in P_q \times [n] \cup \{(a_\emptyset, 1)\}$ such that (1). $\mathsf{DS}[j_c, B, P_q, (b, x), x]$ is *true*, or ($\mathsf{WS}[j_c, B, P_q, (b, x), x]$ is *true* and

($b = a_\emptyset$ or i weakly prefers (a, k) to $(b, x + 1)$)), and (2). $a = a_\emptyset$ or j_c weakly prefers (b, x) to $(a, k + 1)$;

- $P_1, P_2, \ldots, P_{q-1}$ can be assigned to the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ and P_q can be assigned to the subtree $\text{desc}(j_c)$ while the activity a is assigned to x players from the subtrees $\text{desc}(j_1), \text{desc}(j_2), \ldots, \text{desc}(j_{c-1})$ and to $\ell - x$ players from the subtree $\text{ch}(j_c)$, i.e., $\ell \geq 2$, and there exists an $x \in [\ell - 1]$ such that both $TW[j_{c-1}, P_{q-1}, \ell - x]$ and $\mathsf{WS}[j_c, B, P_q \cup \{a\}, (a, k), x]$ are *true*.

We set $TW[j_c, P_q, \ell]$ to *false* otherwise.

A desired assignment exists if and only $TP[j_c, P_q, t - 1]$ (respectively, $TD[j_c, P_q, t - 1]$ and $TW[j_c, P_q, t - 1]$) is *true*. Clearly, the size of each dynamic programming table is at most pn^2 and each entry can be filled in polynomial time. □

A.2.5 Proof of Theorem 3.14

Proof. We provide a parameterized reduction from CLIQUE on regular graphs. Given a regular graph $G = (V, E)$ and a positive integer k, where $|V| = n$, $|E| = m$, and each vertex of G has degree $\delta \geq k - 1$, we create an instance of gGASP whose underlying graph is a clique, as follows.

We create three *vertex activities* $a_i^{(1)}$, $a_i^{(2)}$, and $a_i^{(3)}$ for each $i \in [k]$, one *edge activity* b_j for each $j \in [k(k-1)/2]$, and four other activities d, x, y, and z. For each $v \in V$, we create one *vertex player* v, and for each edge $e = \{u, v\} \in E$, we create two *edge players* e_{uv} and e_{vu}.

Idea. We will create k empty IS-instances $N_i = \{p_1^{(i)}, p_2^{(i)}, p_3^{(i)}\}$ for each $i \in [k]$, and another empty IS-instance $N_g = \{g_1, g_2, g_3\}$ together with the stabilizer g; in order to stabilize these gadgets, each activity $a_i^{(1)}$ should be assigned to some vertex player, and the stabilizer g needs to form a coalition with the players in N_g. Further, we will create dummies of vertex and edge players in such a way that a stable assignment has the following properties:

1. if a vertex player v is assigned to a vertex activity, it forms a coalition of size $\alpha(v)$ that consists of its dummies and $\delta - k + 1$ edge players incident to the vertex; and

2. if an edge player e_{vu} is assigned to an edge activity, it forms a coalition of size $\beta(e)$ that consists of the edge player e_{uv} and the dummies of $\{u, v\}$.

Construction details. Now, let $P = \{j(k + 3) + n \mid j \in [n]\}$, and let $\alpha : V \to P$ be a bijection that assigns a distinct number in P to each vertex $v \in V$. Note that $u \neq v$ implies

that the intervals $[\alpha(u), \alpha(u)+k+1]$ and $[\alpha(v)-1, \alpha(v)+k]$ are disjoint. Similarly, let $Q = \{2j \mid j \in [m]\}$ and let $\beta : E \to Q$ be a bijection that assigns a distinct number in Q to each edge $e \in E$. For each $v \in V$ we construct a set $\mathsf{Dummy}(v)$ of $\alpha(v) - \delta + k - 2$ *dummy vertex players*. Similarly, for each $e \in E$ we construct a set $\mathsf{Dummy}(e)$ of $\beta(e) - 2$ *dummy edge players*.

We will now define the players' preferences.

- Each vertex player $v \in V$ and the players in $\mathsf{Dummy}(v)$ approve each alternative $(a_i^{(1)}, s)$ where $i \in [k]$ and $s \in [\alpha(v), \alpha(v)+k]$.
- Each edge player e_{vu} approves each alternative $(a_i^{(1)}, s)$ where $i \in [k]$ and $s \in [\alpha(v), \alpha(v)+k]$ as well as each alternative in $(b_j, \beta(e))$ where $j \in [k(k-1)/2]$.
- The dummies in $\mathsf{Dummy}(e)$ only approve the alternatives in $(b_j, \beta(e))$ where $j \in [k(k-1)/2]$.

All of these players are indifferent among all alternatives they approve. The stabilizer g approves each alternative of the form $(a_i^{(1)}, s)$ with $i \in [k]$, $s \in \bigcup_{v \in V} [\alpha(v)+2, \alpha(v)+k]$ and is indifferent among them; she also approves $(d, 4)$, but likes it less than all other approved alternatives.

For players in N_g, we have

$$
\begin{aligned}
g_1 &: (d,4) \succ (y,2) \succ (z,1) \succ (x,3) \succ (x,2) \succ (x,1) \succ (a_\emptyset, 1),\\
g_2 &: (d,4) \succ (x,3) \succ (x,2) \succ (z,2) \succ (y,2) \succ (y,1) \succ (a_\emptyset, 1),\\
g_3 &: (d,4) \succ (x,3) \succ (z,2) \succ (z,1) \succ (a_\emptyset, 1).
\end{aligned}
$$

Notice that their preference over the alternatives of x, y, and z is cyclic; hence, in an individually stable assignment, the activity d should be assigned to all the players in N_g together with its stablizer g; otherwise π cannot be stable as we have seen in Example 3.3. Similarly, the preference for players in N_i is cyclic and given by

$$
\begin{aligned}
p_1^{(i)} &: (a_i^{(2)}, 2) \succ (a_i^{(1)}, 1) \succ (a_i^{(3)}, 3) \succ (a_i^{(3)}, 2) \succ (a_i^{(3)}, 1) \succ (a_\emptyset, 1),\\
p_2^{(i)} &: (a_i^{(3)}, 3) \succ (a_i^{(3)}, 2) \succ (a_i^{(1)}, 2) \succ (a_i^{(2)}, 2) \succ (a_i^{(2)}, 1) \succ (a_\emptyset, 1),\\
p_3^{(i)} &: (a_i^{(3)}, 3) \succ (a_i^{(1)}, 2) \succ (a_i^{(1)}, 1) \succ (a_\emptyset, 1).
\end{aligned}
$$

Again, in an individually stable assignment, at least one of the three activities $a_i^{(1)}, a_i^{(2)}, a_i^{(3)}$ must be assigned outside of N_i; the only individual rational way to do this is to assign the activity $a_i^{(1)}$ to other players.

Finally, we take the underlying social network to be a complete graph. Note that the number of activities depends on k, but not on n, and the size of our instance of gGASP is bounded by $O(n^2 + m^2)$.

Correctness. We will now argue that the graph G contains a clique of size k if and only if there exists an individually stable assignment for our instance of gGASP. Suppose that G contains a clique S of size k. We construct an assignment π as follows. We establish a bijection η between S and $[k]$, and for each $v \in V$ we form a coalition of size $\alpha(v)$ that engages in $a_{\eta(v)}$: this coalition consists of v, all players in $\mathsf{Dummy}(v)$, and all edge players e_{vu} such that $u \notin S$. Also, we establish a bijection ξ between the edge set $\{\, e = \{u, v\} \in E \mid u, v \in S \,\}$ and $[k(k-1)/2]$, and assign the activity $b_{\xi(e)}$ to the edge players e_{uv}, e_{vu} as well as to all players in $\mathsf{Dummy}(e)$. Finally, we set $\pi(p_1^{(i)}) = \pi(p_2^{(i)}) = \pi(p_3^{(i)}) = a_i^{(3)}$ for each $i \in [k]$ and $\pi(g) = \pi(g_1) = \pi(g_2) = \pi(g_3) = d$, and assign the void activity to the remaining players. We will now argue that the resulting assignment π is individually stable.

Clearly, no player assigned to an activity $a_i^{(1)}$ or b_j wishes to deviate. Now, consider a player $v \in N$ with $\pi(v) = a_\emptyset$; by construction, v only wants to join a coalition if it engages in an activity $a_i^{(1)}$ and its size is in the interval $[\alpha(v) - 1, \alpha(v) + k]$, and no such coalition exists. The same argument applies to players in $\mathsf{Dummy}(v)$. Similarly, consider an edge player e_{vu} with $\pi(e_{vu}) = a_\emptyset$. We have $v \notin S$, and therefore e_{vu} does not want to join any of the existing coalitions; the same argument applies to all dummies of $e = \{u, v\}$. Further, players in N_i do not want to deviate since players $p_2^{(i)}$ and $p_3^{(i)}$ are allocated to their top alternative and the activity $a_i^{(1)}$ is assigned to at least one player. Also, players in N_g do not want to deviate since they are allocated one of their top choices. Finally, the stabilizer g does not want to deviate, since there is no coalition of size $s \in \bigcup_{v \in V} [\alpha(v) + 1, \alpha(v) + k - 1]$ that engages in an activity $a_i^{(1)}$. Hence, π is individually stable.

Conversely, suppose that there exists an individually stable feasible assignment π. Notice that π cannot allocate an activity $a_i^{(1)}$ to players in N_i, or leave it unallocated, since no such assignment can be individually stable. Thus, each vertex activity $a_i^{(1)}$ is allocated to a coalition whose size lies in the interval $[\alpha(v), \alpha(v) + k]$ for some $v \in V$. Further, individual stability implies that π allocates the activity d to the players in N_g and its stabilizer g. Now, if some vertex activity $a_i^{(1)}$ is assigned to s players, where $s \in [\alpha(v) + 1, \alpha(v) + k - 1]$ for some $v \in V$, the stabilizer g would then deviate to that coalition. Further, for each $v \in V$, the number of players other than g who approve the alternatives of the form $(a_i^{(1)}, \alpha(v) + k)$ is at most $\alpha(v) + k - 1$ but g is assigned to d; thus, for each $a_i^{(1)}$ we have $|\pi^{a_i^{(1)}}| = \alpha(v)$ for some $v \in V$. Consider a player $v \in S$, and let $a(v)$ be the activity assigned to $\alpha(v)$ players under π. We have $\pi(v) = a(v)$, since otherwise v has an IS-deviation to $a(v)$.

Now, let $S = \{v \in V \mid \pi(v) = a_i^{(1)} \text{ for some } i \in [k]\}$. By construction, $|S| = k$. We can show that S is a clique in a similar manner to the proof of Claim 3.2. □

A.2.6 Proof of Theorem 3.17

Proof. Our problem is in NP by Proposition 3.1. We again reduce from X3C. Let $(V, \mathcal{S})$ be an instance of X3C where $V = \{v_1, v_2, \ldots, v_{3k}\}$ and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$.

We define the set of activities by $A^* = \{a, b, c_1, c_2\}$. We introduce three players x_1, x_2, x_3 and one player S_j for each $S_j \in \mathcal{S}$.

Idea. Again, we will create the dummies of each $v_i \in V$ and the preferences of players as follows:

- The players x_1, x_2, and x_3 form an empty core instance. In a stable assignment, they need to be assigned to the activity a.
- If the dummies of v_i are assigned to the void activity and none of the players corresponding to the sets including v_i (denoted by $\mathcal{S}(v_i)$) is assigned to the activity c_2, then these players will form a blocking coalition of size $\beta(v_i)$ and deviate to activity b.

Construction details. For each $v_i \in V$, we let $\mathcal{S}(v_i) = \{\, S_j \mid v_i \in S_j \in \mathcal{S} \,\}$ and $\beta(v_i) = i + 3k + 1$ and create a set $\mathsf{Dummy}(v_i) = \{d_i^{(1)}, d_i^{(2)}, \ldots, d_i^{(\beta(v_i)-|\mathcal{S}(v_i)|)}\}$ of dummy players. For each $i \in [3k]$ the number $\beta(v_i)$ is the target coalition size when all players in $\mathcal{S}(v_i)$ are engaged in activity b, together with the players in $\mathsf{Dummy}(v_i)$. We then create an edge between any pair of players.

The agents' preferences over alternatives are defined as follows. For each $S_j \in \mathcal{S}$, we let $B_j = \{b\} \times \{\, \beta(v_i) \mid v_i \in S_j \,\}$. The preferences of each player $S_j \in \mathcal{S}$ are given by

$$S_j : (c_2, k) \succ B_j \succ (c_1, n-k) \succ (a_\emptyset, 1).$$

For each $v_i \in V$ the dummy players in $\mathsf{Dummy}(v_i)$ only approve the alternative $(b, \beta(v_i))$.

Finally, the preferences of players x_1, x_2, and x_3 are given by

$$\begin{aligned} x_1 &: (c_1, 2) \sim (c_2, 2) \succ (a, 3) \succ (a_\emptyset, 1) \\ x_2 &: (a, 2) \succ (c_1, 2) \sim (c_2, 2) \succ (a, 3) \succ (a_\emptyset, 1) \\ x_3 &: (a, 3) \succ (c_1, 1) \sim (c_2, 1) \succ (a, 2) \succ (a_\emptyset, 1). \end{aligned}$$

Notice that the preferences of x_1, x_2, and x_3, when restricted to $\{a, c_1, a_\emptyset\} \times [1, 2, 3]$ or $\{a, c_2, a_\emptyset\} \times [1, 2, 3]$, form an empty core instance (Example 3.2).

We will show that $(V, \mathcal{S})$ contains an exact cover if and only if there exists a core stable feasible assignment.

Correctness. Suppose that $(V, \mathcal{S})$ admits an exact cover $\mathcal{S}'$. Then, we construct a feasible assignment π by setting $\pi(x_i) = a$ for $i = 1, 2, 3$, $\pi(S_j) = c_2$ for $S_j \in \mathcal{S}'$, $\pi(S_j) = c_1$ for $S_j \in \mathcal{S} \setminus \mathcal{S}'$, and assigning the remaining players to the void activity. Clearly, no subset together with c_i $(i = 1, 2)$ or a strongly blocks π. We will show that no subset T together with activity b strongly blocks π. Suppose towards a contradiction that such a subset T exists; as no players in $\{x_1, x_2, x_3\}$ approves alterntives of b, it must be the case that $|T| = \beta(v_i)$ for some $v_i \in V$ and hence T consists of agents who approve $(b, \beta(v_i))$, i.e., $T = \mathcal{S}(v_i) \cup \mathsf{Dummy}(v_i)$ for some $v_i \in V$. However, since $\mathcal{S}'$ is an exact cover, there is an agent $S_j \in \mathcal{S}' \cap \mathcal{S}(v_i)$ with $\pi(S_j) = c_2$, and this agent prefers (c_2, k) to $(b, \beta(v_i))$, a contradiction. Hence, π is core stable.

Conversely, suppose that there exists a core stable feasible assignment π and let $\mathcal{S}' = \{\, S_j \in \mathcal{S} \mid \pi(S_j) = c_2 \,\}$. We will show that $\mathcal{S}'$ is an exact cover. Observe that by individual rationality, the only agents who can be allocated to a are the players x_1, x_2, and x_3. Hence, by core stability, both activities c_1 and c_2 must be allocated outside of the players x_1, x_2, and x_3; otherwise, no core stable outcome would exist as we have seen in Example 3.2. The only indivudally rational way to do this is to assign activity c_2 to k players from $\mathcal{S}$, and assign activity c_1 to the remaining $n - k$ players in $\mathcal{S}$. Thus, $|\mathcal{S}'| = k$. Then, no player in $\mathcal{S}$ can be engaged in activity b, and hence, by individual rationality, all dummy players must be assigned to the void activity. Now suppose towards a contradiction that $\mathcal{S}'$ is not a cover, i.e., there exists an element $v_i \in V$ such that $\mathcal{S}(v_i) \cap \mathcal{S}' = \emptyset$. This would mean that π assigns all players in $\mathcal{S}(v_i)$ to the activity c_1, and hence the coalition $\mathcal{S}(v_i) \cup \mathsf{Dummy}(v_i)$ together with the activity b strongly blocks π, contradicting the stability of π. □

A.2.7 Proof of Theorem 3.19

Proof. We describe a parameterized reduction from the W[1]-hard MULTICOLORED CLIQUE problem. Given an undirected graph $G = (V, E)$, a positive integer $h \in \mathbb{N}$, and a vertex coloring $\phi \colon V \to [h]$, MULTICOLORED CLIQUE asks whether G admits a colorful h-clique, that is, a size-h vertex subset $Q \subseteq V$ such that the vertices in Q are pairwise adjacent and have pairwise distinct colors. Without loss of generality, we assume that there are exactly q vertices of each color for some $q \in \mathbb{N}$, and that there are no edges between vertices of the same color.

Let (G, h, ϕ) be an instance of MULTICOLORED CLIQUE with $G = (V, E)$. For convenience, we write $V^{(i)} = \{v_1^{(i)}, \ldots, v_q^{(i)}\}$ to denote the set of vertices of color i for every $i \in [h]$. We construct our gGASP instance as follows. We have one *vertex activity* v for each vertex $v \in V$, one *edge activity* e for each edge $e \in E$, three *color activities* a_i, b_i,

and c_i for each color $i \in [h]$, three *colorpair activities* $a_{\{ij\}}$, $b_{\{ij\}}$, and $c_{\{ij\}}$ for each color pair $\{i, j\} \subseteq [h], i \neq j$, and four other additional activities d, x, y, and z.

Idea. We will have one *color gadget* $\mathsf{Color}(i)$ for each color $i \in [h]$, one *colorpair gadget* $\mathsf{ColorPair}(\{i, j\})$ for each color pair $\{i, j\}, i \neq j$, and one empty IS-instance $N_g = \{g_1, g_2, g_3\}$ together with the stabilizer g. For most of the possible assignments, these gadgets will be unstable, unless the following holds:

1. For each color $i \in [h]$ the first two players from the color gadget select a vertex of color i (by being assigned together to the corresponding vertex activity).
2. For each color pair $\{i, j\} \subseteq [h], i \neq j$ the first two players of the colorpair gadget select an edge connecting one vertex of color i and one vertex of color j (by being assigned together to the corresponding edge activity).
3. Every selected edge for the color pair $\{i, j\}$ must be incident to both vertices selected for color i and color j.
4. The stabilizer g as well as the players in N_g are engaged in the activity d.

If the four conditions above hold, then the assignment must encode a colorful h-clique.

Construction details. The color gadget $\mathsf{Color}(i)$, $i \in [h]$ consists of the following three players.

$p_1^{(i)} : (v_1^{(i)}, 2) \succ E(v_1^{(i)}) \times [3, 5] \succ (v_2^{(i)}, 2) \succ E(v_2^{(i)}) \times [3, 5] \succ \cdots \succ (v_q^{(i)}, 2) \succ E(v_q^{(i)}) \times [3, 5] \succ (b_i, 2) \succ (a_i, 1) \succ (c_i, 3) \succ (c_i, 2) \succ (c_i, 1) \succ (a_\emptyset, 1),$

$p_2^{(i)} : (v_q^{(i)}, 2) \succ E(v_q^{(i)}) \times [3, 5] \succ (v_{q-1}^{(i)}, 2) \succ E(v_{q-1}^{(i)}) \times [3, 5] \succ \cdots \succ (v_1^{(i)}, 2) \succ E(v_1^{(i)}) \times [3, 5] \succ (c_i, 3) \succ (c_i, 2) \succ (a_i, 2) \succ (b_i, 2) \succ (b_i, 1) \succ (a_\emptyset, 1),$

$p_3^{(i)} : (c_i, 3) \succ (a_i, 2) \succ (a_i, 1) \succ (a_\emptyset, 1),$

where $E(v)$ to denote the set of activities corresponding to edges incident to vertex v for every $v \in V$.

The colorpair gadget $\mathsf{ColorPair}(\{i, j\})$, $\{i, j\} \subseteq [h], i \neq j$, consists of three players.

$p_1^{\{i,j\}} : E(\{i, j\}) \times [2, 5] \succ (b_{\{ij\}}, 2) \succ (a_{\{ij\}}, 1) \succ (c_{\{ij\}}, 3) \succ (c_{\{ij\}}, 2) \succ (c_{\{ij\}}, 1) \succ (a_\emptyset, 1)$

$p_2^{\{i,j\}} : E(\{i, j\}) \times [2, 5] \succ (c_{\{ij\}}, 3) \succ (c_{\{ij\}}, 2) \succ (a_{\{ij\}}, 2) \succ (b_{\{ij\}}, 2) \succ (c_{\{ij\}}, 1) \succ (a_\emptyset, 1),$ and

$p_3^{\{i,j\}} : (c_{\{ij\}}, 3) \succ (a_{\{ij\}}, 2) \succ (a_{\{ij\}}, 1) \succ (a_\emptyset, 1)$

where $E(\{i, j\})$ denotes the set of edge activities incident to vertices of color i and j.

There is a stabilizer player g with the following preferences.

$$g : E \times [4,5] \succ (d,4) \succ (a_\emptyset, 1).$$

The set N_g consists of three players.

$$g_1 : (d,4) \succ (y,2) \succ (z,1) \succ (x,3) \succ (x,2) \succ (x,1) \succ (a_\emptyset, 1),$$
$$g_2 : (d,4) \succ (x,3) \succ (x,2) \succ (z,2) \succ (y,2) \succ (y,1) \succ (a_\emptyset, 1),$$
$$g_3 : (d,4) \succ (x,3) \succ (z,2) \succ (z,1) \succ (a_\emptyset, 1).$$

We take the underlying social network to be a complete graph. Together, there are $3h + 3\binom{h}{2} + 4$ players, namely

$$N = \bigcup_{i \in [h]} \mathsf{Color}(i) \cup \bigcup_{i \neq j \in [h]} \mathsf{ColorPair}(\{i,j\}) \cup \{g\} \cup N_g.$$

Note that the number of players depends on h, but not on n, and the size of our instance of gGASP is bounded by $O(n^2 + m^2)$.

Correctness. We will now argue that the graph G admits a colorful clique of size h if and only if our instance of gGASP admits a Nash stable feasible assignment or an individually stable feasible assignment.

Suppose that there exists a colorful clique H of size h. We construct a Nash stable assignment π where

- the first two players $p_1^{(i)}$ and $p_2^{(i)}$ of the color gadget $\mathsf{Color}(i)$ are assigned to the activity corresponding to the vertex of color i from H,
- the last player $p_3^{(i)}$ of the color gadget $\mathsf{Color}(i)$ is assigned to the activity a_i,
- the first two players $p_1^{\{i,j\}}$ and $p_2^{\{i,j\}}$ of the colorpair gadget $\mathsf{ColorPair}(\{i,j\})$ are assigned to the activity corresponding to the edge between the vertices of color i and j in H,
- the last player $p_3^{\{i,j\}}$ of the colorpair gadget $\mathsf{ColorPair}(\{i,j\})$ is assigned to the activity $a_{\{ij\}}$, and
- the stabilizer g and the players in N_g are assigned to the activity d.

Observe that for a successful Nash deviation a player must join an existing non-empty coalition, because no player prefers a size-one activity to the currently assigned one. By construction, the last player of each color gadget and the last player of the colorpair gadget

cannot deviate (no other players engage in an approved activity). Consider the first two players of a color gadget $\mathsf{Color}(i)$. They cannot deviate to a vertex activity, because their current activity is their only approved vertex activity that has some players assigned to it. They cannot deviate to an edge activity either, because they would only prefer edge activities corresponding to edges that are not incident to the vertex of color i; these activities, however, have no players assigned to them. The first two players of each colorpair gadget $\mathsf{ColorPair}(\{i,j\})$ and the players in N_g do not wish to deviate since they are each assigned to their top alternative. Finally, the stabilizer g does not wish to deviate since there is no coalition of size 3 or 4 assigned to an edge activity. Thus, we have a Nash stable feasible assignment, and hence, an individually stable assignment.

Conversely, suppose that there exists an individually stable feasible assignment π. Then, π must assign the activity d to the stabilizer as well as the players in N_g; otherwise, π cannot be stable as we have seen in Example 3.3. Likewise, the first two players of each colorpair gadget $\mathsf{ColorPair}(\{i,j\})$ must be assigned to the same edge activity corresponding to some edge $\hat{e}_{i,j} \in E$. We say that these players "select edge $\hat{e}_{i,j}$".

Now suppose towards a contradiction that the first player $p_1^{(i)}$ of color gadget $\mathsf{Color}(i)$ is assigned to an edge activity $\hat{e}_{i,j}$ incident to vertices of color i and j. Then, such a coalition has size in $[3,5]$ by individual rationality, and must not contain the stabilizer g, because g has to be engaged in activity d as discussed before. Hence the player $p_1^{(i)}$ can only be assigned to $\hat{e}_{i,j}$ together with $p_2^{(i)}$ and the first two players from the colorpair gadget $\mathsf{ColorPair}(\{i,j\})$, which results in a coalition of size 3 or 4. This would, however, cause an IS-deviation by the stabilizer g to the edge activity $\hat{e}_{i,j}$, a contradiction. The same argument applies to when the second player $p_2^{(i)}$ is assigned to an edge activity. Therefore, the first two players of each color gadget $\mathsf{Color}(i)$ are assigned to the same vertex activity corresponding to some vertex $v_{\ell_i} \in V^{(i)}$. We say that these players "select vertex v_{ℓ_i}". (Note that at this point, the coalition structure of any stable outcome is already fixed: The first two players of each color gadget and of each colorpair gadget must form a coalition of size two, respectively, and all other players must be form singleton coalitions.)

Now, assume towards a contradiction that the selected vertices and edges do not form a colorful clique of size h. The size and colorfulness are clear from the construction. Hence, there must be some pair $\{v_{\ell_i}, v_{\ell_j}\}$ of selected vertices that are not adjacent. However, this would imply that colorpair gadget $\mathsf{ColorPair}(\{i,j\})$ selected an edge that is not incident to vertex v_{ℓ_i} or not incident to vertex v_{ℓ_j}. Without loss of generality let it be non-incident to vertex v_{ℓ_i}. Now, there are two cases. First, if $\hat{e}_{i,j} \in E(v_x^{(i)})$ with $x < \ell_i$, then player $p_1^{(i)}$ would have an IS-deviation to the activity corresponding to $\hat{e}_{i,j}$. Second, if $\hat{e}_{i,j} \in E(v_x^{(i)})$ with $x > \ell_i$, then player $p_2^{(i)}$ would have an IS-deviation to the activity corresponding to $\hat{e}_{i,j}$.

In both cases we have a contradiction to the assumption that π is individually stable. A similar argument applies to the case when there is a Nash stable assignment. □

A.3 Appendix: Chapter 5

A.3.1 Proof of Theorem 5.4

Proof. The proof is similar to the one for Theorem 5.5. Again, we reduce from EXACT-3-COVER (X3C). Recall that an instance of X3C is given by a set of elements $X = \{x_1, x_2, \ldots, x_{3s}\}$ and a family $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ of three-element subsets of X; it is a 'yes'-instance if and only if X can be covered by s sets from $\mathcal{S}$.

In what follows, we say that a valid allocation π is *perfect* if each player receives the best connected bundle, i.e., $u_i(\pi(i)) = \max\{\, u_i(X) \mid X \subseteq V \ \wedge\ X \text{ is connected} \,\}$ for all $i \in N$.

Consider an instance $(X, \mathcal{S})$ of X3C. We write the *frequency* of each $x \in X$ as $p_x = |\{\, S \in \mathcal{S} \mid x \in S \,\}|$. For each $S \in \mathcal{S}$, we denote the three elements of S by x_S^1, x_S^2, x_S^3. We construct an instance I of CFD as follows. For each set $S \in \mathcal{S}$, we create a path of three vertices x_S^1, x_S^2, x_S^3 with the center x_S^2; for each $k \in [s]$, we create a path of three dummy vertices d_k^1, d_k^2, d_k^3 with the center d_k^2. See Figure A.2 for an illustration.

Figure A.2: Graph constructed in the proof of Theorem 5.4.

For each set $S \in \mathcal{S}$, we create one player i_S who has non-zero utility $1/(3s+3)$ only for each element x_S^1, x_S^2, x_S^3 contained in the set, and each dummy d_k^h for $k \in [s]$ and $h \in [3]$; for each element $x \in X$ we create one player i_x who has non-zero utility $1/p_x$ only for the vertices x_S^h with $x_S^h = x$. Note that by construction, each player i_S ($S \in \mathcal{S}$) can receive a bundle of value at most $1/(s+1)$, while each player i_x ($x \in X$) can receive a bundle of value at most $1/p_x$. By construction, $u_i(V) = 1$ for each $i \in N$.

We will first show that I admits a perfect valid allocation if and only if there is an exact cover.

Suppose that there is a cover $\mathcal{S}'$ of size s. Then, one can construct a perfect valid allocation π as follows.

- Each player i_S whose corresponding set S appears in the cover $\mathcal{S}'$ is arbitrarily assigned to one of the the dummy triples d_k^1, d_k^2, d_k^3 for $k \in [s]$.

- Each player i_S whose corresponding set S does not appear in the cover $\mathcal{S}'$ is assigned to the triple x_S^1, x_S^2, x_S^3.
- Each player i_x is assigned to the element vertex x_S^k such that $x = x_S^k$ and $S \in \mathcal{S}'$.

Each player is assigned to a connected piece of maximum value, implying that π is a perfect valid allocation.

Conversely, suppose that there exists a valid allocation such that each player i_S receives a bundle of value $1/(s+1)$ and each player i_x receives a bundle of value $1/p_x$. Now, let $\mathcal{S}' \subseteq \mathcal{S}$ be the family of the sets $S \in \mathcal{S}$ where one of its elements x_S^1, x_S^2, x_S^3 is allocated to some x-players i_x. Since each player i_x needs to be allocated to an item for which she has a positive value, $\mathcal{S}'$ forms a cover for X and hence $|\mathcal{S}'| \geqslant s$. On the other hand, as the number of dummy triples is s, there must be at least $|\mathcal{S}| - s$ players i_S who need to be allocated to triple intervals x_S^1, x_S^2, x_S^3. Thus, $\mathcal{S}'$ has size at most s and constitutes an exact cover for X.

Now, suppose that π is a Pareto-optimal valid allocation in this instance. It follows that, by looking at a Pareto-optimal allocation and the utility that each player enjoys, we can decide whether the given instance has an exact cover. □