Statistically Stable Communities with Limited Interactions

Ayumi Igarashi, Jakub Sliwinski and Yair Zick

Abstract

A community needs to be partitioned into disjoint groups; each community member has an underlying preference over the groups that they would want to be a member of. We are interested in finding a stable community structure: one where no subset of members S wants to deviate from the current structure. We model this setting as a hedonic game, where players are connected by an underlying interaction network, and can only consider joining groups that are connected subgraphs of the underlying graph. We analyze the relation between network structure, and one's capability to infer statistically stable (also known as PAC stable) player partitions from data. We show that when the interaction network is a forest, one can efficiently infer PAC stable coalition structures. Furthermore, when the underlying interaction graph is not a forest, efficient PAC stabilizability is no longer achievable. Thus, our results completely characterize when one can leverage the underlying graph structure in order to compute PAC stable outcomes for hedonic games. Finally, given an unknown underlying interaction network, we show that it is NP-hard to decide whether there exists a forest consistent with data samples from the network.

1 Introduction

A professor wants her students to complete a group programming project. In order to do so, students should divide into project groups with a few students in each; naturally, some groups will be objectively better than others. However, students seldom try to find a group that's objectively optimal for them; they would rather join groups that have at least one or two of their friends. This type of scenario falls into the realm of *constrained coalition formation*; in other words, how should we partition a group of people given that (a) they have preferences over the groups they are assigned to and (b) they have limited interactions with one another? Other scenarios fitting this description include

(a) Seating arrangements at a wedding (or at conference banquets): some guests should absolutely not be seated together, while others would probably enjoy one another's company. However, it should always be the case that every guest has at least one acquaintance seated at their table.

(b) Group formation on social media: given a social media network (e.g. Facebook), people prefer being affiliated with certain groups; however, they are limited to joining groups that already contain their friends.

Constrained coalition formation problems are often modeled as *hedonic games*. Hedonic games formally capture a simple, yet compelling, paradigm: how does one partition players into groups, while factoring individual players' preferences? The literature on hedonic games is primarily focused on finding "good" *coalition structures* — partitions of players into disjoint groups. A set of coalition structures satisfying certain desiderata is called a *solution concept*. A central hedonic solution concept is *coalitional stability*: given a coalition structure π, we say that a set of players (also known as a *coalition*) S can deviate from π if every $i \in S$ prefers S to its assigned group under π; a coalition structure π is stable if no coalition S can deviate. In other words, S contains at least one player i who prefers its current coalition

(denoted $\pi(i)$) to S. The set of stable coalition structures — also known as the *core* of the hedonic game — may be empty; what's worse, even when it is known to be non-empty, finding a stable coalition structure may be computationally intractable. Moreover, in the cases where there exist efficient algorithms computing coalition structures in the core of the hedonic game, these algorithms often assume full knowledge of the underlying hedonic game; that is, in order to work, the algorithm needs to have either *oracle access to player preferences* (i.e. queries of the form 'does player i prefer coalition S to coalition T?'), or *structural knowledge of the underlying preference structure* (e.g. some concise representation of player preferences that one can leverage in order to obtain a poly-time algorithm).

Neither assumption is realistic in practice: eliciting user preferences is notoriously difficult, especially over combinatorially complex domains such as subsets of players. If one forgoes preference elicitation and opts for mathematically modeling preferences (e.g. assuming that users have additive preferences over coalition members), it is not entirely obvious what mathematical model of user preferences is valid. This leads us to the following natural question: can we find a stable coalition structure when player preferences are unknown? Recent works [5, 8, 40] propose a statistical approach to stability in collaborative environments. In this framework, one assumes the existence of user preference data over some coalitions, which is then used to construct *probably approximately stable* outcomes (the notion is referred to as PAC stability). In this paper, we explore the relation between structural assumptions on player preferences, and computability of PAC stable outcomes.

1.1 Our Contributions

We assume that there exists some underlying *interaction network* governing player preferences; that is, there exists some graph whose nodes are players, and only connected coalitions are feasible. Within this framework, we show that if player preferences are restricted by a forest, one can compute a PAC stable outcome using only a polynomial number of samples. Surprisingly, even if the underlying forest structure is not known to the learner, PAC stabilizability still holds, despite the fact that it may be computationally intractable to find an approximate forest structure that is likely consistent with the true interaction graph. In contrast, we show that it is impossible to find a PAC stable outcome even if the graph contains a single cycle. The latter result is constructive: we show that whenever the underlying interaction graph does contain a cycle, one can construct a sample distribution for which it would be impossible to elicit a PAC stable outcome.

Our positive result for forests is interesting in several respects. First, while one can find PAC stable outcomes in polynomial time, computing stable outcomes for hedonic games on forests is computationally intractable [27]; second, unlike Sliwinski and Zick, we do not require that player 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 problem we model, which assumes ordinal information about player preferences, rather than cardinal utilities. Finally, in Section 5, we prove a non-trivial technical result on learning forest structures that is of independent interest. Briefly, we study the following problem: we are given samples of graph vertices, each labeled either 'connected' or 'disconnected'; we need to decide whether there exists some forest T^* that is consistent with the sample — i.e. all connected sets of vertices are connected under T^* and all disconnected sets are not. We show that when all of our vertex samples are connected (i.e. we do not observe any disconnected components), it is possible to efficiently learn an underlying forest structure (if one exists); on the other hand, if one assumes that both connected and disconnected sets are presented to the learner, it is computationally intractable to decide whether there exists a forest, or even a path, that is consistent with the samples.

1.2 Related Work

There exists a rich body of literature studying hedonic games from an economic perspective (e.g. [9, 11, 21]). More recently, the AI community has begun studying both computational and analytical properties of hedonic games (see e.g. [2, 14, 18, 24, 37], and [3, 42] for an overview). Interaction networks in cooperative games were first introduced by Myerson [35]. The relation between graph structure and stability in the classic cooperative game setting is also relatively well-understood. Demange [19] shows that if the underlying interaction network is a forest, then the core is not empty; further studies [13, 32] establish relations between approximate stability and the underlying graph structure, while Chalkiadakis et al. [15] study the computational complexity of finding core outcomes in graph restricted environments. Igarashi and Elkind [27] establish both the existence of stable coalition structures in hedonic games over forests, as well as the computational intractability of finding stable coalition structures; Peters [36] studies the relation between hedonic solution concepts and the treewidth of the underlying interaction graph.

Several works study learning based game-theoretic solution concepts. Sliwinski and Zick [40] 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 [5, 8], rankings [7], auctions [4, 6, 33, 34] and noncooperative games [22, 39].

2 Preliminaries

Throughout this paper, vectors are referred to by the notation $\vec{x}$, sets are denoted by uppercase letters; given a value $s \in \mathbb{N}$, we set $[s] = \{1, \ldots, s\}$. A *hedonic game* is given by a pair $\langle N, \succeq \rangle$, where $N = [n]$ is a finite set of *players*, and $\succeq = (\succeq_1, \ldots, \succeq_n)$ is a list of preferences players in N have over subsets of N (also referred to as *coalitions*); in more detail, for every $i \in N$, we write $\mathcal{N}_i = \{ S \subseteq N \mid i \in S \}$; $\succeq_i$ describes a complete and transitive preference relation over $\mathcal{N}_i$. For each $i \in N$, let $\succ_i$ denote the strict preference derived from $\succeq_i$, i.e., $S \succ_i S'$ if $S \succeq_i S'$, but $S' \not\succeq_i S$. An outcome of a hedonic game is a *coalition structure*, i.e., a partition π of N into disjoint coalitions; we denote by $\pi(i)$ the coalition containing $i \in N$. A solution concept is a mapping whose input is a hedonic game $\langle N, \succeq \rangle$, and whose output is a (possibly empty) set of coalition structures. The *core* is the most fundamental solution concept in hedonic games. First, we say that a coalition S *strongly blocks* a coalition structure π if every player $i \in S$ strictly prefers S to its current coalition $\pi(i)$, i.e. $S \succ_i \pi(i)$. A coalition structure π is said to be *core stable* if no coalition $S \subseteq N$ strongly blocks π.

2.1 Interaction Networks

Given an undirected graph $G = \langle N, E \rangle$ whose nodes are the player set, we restrict the space of *feasible coalitions* to be the set of connected subsets of G; we denote by $\mathcal{F}_E$ the set of feasible coalitions. Intuitively, we restrict our attention to coalition structures where all group members form a social subnetwork of the underlying interaction graph. Note that when G is a clique, all coalitions are feasible, and the result is a standard (unrestricted) hedonic game. From now on, we define a *hedonic graph game* as the tuple $\langle N, \succeq, E \rangle$; here, N is the player set, $\succeq$ their preference relations, and E the edges of the underlying interaction network. We focus our attention only on core stable coalition structures that consist of feasible coalitions.

In what follows, it is useful to express player 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 $\langle N, \mathcal{V} \rangle$ where $\mathcal{V}$ is a collection of functions $v_i : \mathcal{N}_i \to \mathbb{R}$ for each $i \in N$.

This representation allows us to seamlessly integrate ideas from PAC learning into the hedonic games model, and is indeed quite common in other works studying hedonic games. However, as we later show, our main result (Theorem 4.1) still holds when we transition from a utility-based cardinal model, to a preference-based ordinal model.

2.2 PAC Learning

We provide a brief introduction to PAC learning[1]. The basic idea is as follows: we are given an unknown function $v : 2^N \to \mathbb{R}$ (a *target concept* in the language of PAC learning) that assigns values to subsets of players. In addition, we are given a set of m samples $(S_1, v(S_1)), \ldots, (S_m, v(S_m))$ where $S_j \subseteq N$ and $v(S_j)$ is the valuation of v over S_j; we wish to estimate v on subsets we did not observe. We assume that v belongs to a *hypothesis class* $\mathcal{H}$ (say, we know that v is an additive valuation). Note that this assumption is necessary if we want to make any educated guess regarding the possible values v takes. Without this assumption, the estimate

$$v^*(S) = \begin{cases} v(S) & \text{if } S \in \{S_1, \ldots, S_m\} \\ 0 & \text{otherwise.} \end{cases}$$

is as good a guess as any. 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 $\epsilon, \delta > 0$, $\mathcal{H}$ 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 (though unlikely) that we draw the same subset in every one of our m samples. 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* (see e.g. Chapter 11 of [1]). 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 $v(S_j) \geq r_j$ iff $\ell_j = 1$. The *pseudo-dimension* of $\mathcal{H}$, denoted $P_{dim}(\mathcal{H})$ 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.

[1] What we show here is but one of many variants on the theory of PAC learning. There are many excellent sources on this classic theory; we refer our reader to [1, 28, 38]

Theorem 2.1 (Anthony and Bartlett [1]). *A class of functions $\mathcal{H}$ is efficiently (ϵ, δ) PAC learnable using $m = poly(P_{dim}(\mathcal{H}), \frac{1}{\epsilon}, \log \frac{1}{\delta})$ 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 m. Furthermore, if $P_{dim}(\mathcal{H})$ is superpolynomial in n, $\mathcal{H}$ is not 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.

2.3 PAC Stabilizability

When studying hedonic games, one is not necessarily interested in eliciting approximately accurate user preferences over coalitions using data; in our case, we are interested in identifying core stable coalition structures. Intuitively, it seems that the following idea might work: first, infer player utilities from data and obtain a PAC approximation of the original hedonic game; next, find a coalition structure that stabilizes the approximate hedonic game. This approach, however, may be overcomplicated: first, it may be impossible to PAC learn player preferences from data (this depends on the hypothesis class); moreover, computing a core coalition structure for the learned game may be computationally intractable. Sliwinski and Zick [40] propose learning a stable outcome directly from data. They introduce a statistical notion of core stability for hedonic games, which they term *PAC stability* (this term was first used by Balcan et al. [5] for cooperative transferable utility games).

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 $\langle N, \mathcal{V} \rangle$ 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 $\mathcal{A}$ *PAC stabilizes* $\mathcal{H}$, if for any hedonic game $\langle N, \mathcal{V} \rangle \in \mathcal{H}$, distribution $\mathcal{D}$ over 2^N, and parameters $\epsilon, \delta > 0$, with probability $\geq 1 - \delta$, $\mathcal{A}$ outputs an ϵ-PAC stable coalition structure if it exists; again, if the running time of the algorithm $\mathcal{A}$ and the number of samples, m, are bounded by a polynomial in n, $\frac{1}{\epsilon}$ and $\log \frac{1}{\delta}$, then we say that $\mathcal{A}$ *efficiently PAC stabilizes* $\mathcal{H}$. Similarly, we say that $\mathcal{H}$ is *(efficiently) PAC stabilizable* if there is some algorithm $\mathcal{A}$ that (efficiently) PAC stabilizes $\mathcal{H}$.

3 Learning Hedonic Graph Games

In what follows we consider the following hypothesis class.

Definition 3.1. *For an undirected graph $G = \langle N, E\rangle$, let $\mathcal{H}_G$ be the class of all hedonic games $\langle N, \mathcal{V}\rangle$ where for each player $i \in N$, $v_i(\{i\}) = 0$ and player i strictly prefers its singleton to any disconnected coalition $S \in \mathcal{N}_i \setminus \mathcal{F}_E$, i.e., $v_i(S) < 0$ for all $S \in \mathcal{N}_i \setminus \mathcal{F}_E$.*

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 3.2 shows, even when we are given additional information about the underlying graph interaction network, players' preferences are not PAC learnable.

Theorem 3.2. *For any graph $G = \langle N, E\rangle$ with exponentially many connected coalitions, the class $\mathcal{H}_G$ is not efficiently PAC learnable.*

Proof. Recall that $\mathcal{F}_E$ is the set of all feasible coalitions over $G = \langle N, E\rangle$; by assumption, $|\mathcal{F}_E|$ 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 $S \in \mathcal{F}_E$, based on other feasible coalitions' utilities. This immediately implies that the set $\mathcal{F}_E$ can be pseudo-shattered by $\mathcal{H}_i$. Hence $P_{dim}(\mathcal{H}_i)$ is at least exponential, and by Theorem 2.1, $\mathcal{H}_i$ is not efficiently PAC learnable. □

As an immediate corollary, forest interaction structures do not admit PAC learnable preference structures in general; this is true even if G is a star graph over n players, since the number of feasible coalitions is exponential in n.

Corollary 3.3. *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 3.2, 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 [27]. However, we now show how one can still exploit the structural properties of forest graph structures to efficiently compute PAC stable outcomes.

4 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 G is known to us; in other words, we know that our game belongs to the hypothesis class $\mathcal{H}_G$. In Section 5, we show how one can forgo this assumption.

Theorem 4.1. *If $G = \langle N, E\rangle$ is a forest, $\mathcal{H}_G$ is efficiently PAC stabilizable.*

Proof. We claim that Algorithm 1 PAC stabilizes $\mathcal{H}_G$. It is related to the algorithm introduced in Demange [19] used to find core stable outcomes for forest-restricted hedonic games[2] in the full information setting. Intuitively, instead of identifying the *guaranteed coalition* for each player precisely, Algorithm 1 approximates it. If the input graph $\langle N, E\rangle$ is a forest, we can process each of its connected components separately, so we can assume that $\langle N, E\rangle$ is a tree.

We first provide an informal description of our algorithm, followed by pseudocode. The algorithm first transforms $\langle N, E\rangle$ into a rooted tree with root r by orienting the edges in E towards the leaves. For every player i starting from the bottom to the top, the algorithm identifies B_i - a coalition containing i, the best for i observed in the samples that is entirely contained in i's subtree, such that others in B_i prefer it to their own best guaranteed coalition; in other words, $B_i \succeq_j B_j$ for all $j \in B_i$. Having identified B_i for every $i \in N$, players are partitioned according to the B_i's from top-down. The main concern is to ensure that B_i is a good approximation of its full-information counterpart; this is guaranteed by taking a sufficiently large sample size $m = \lceil \frac{n}{\epsilon} \log \frac{n}{\delta} \rceil$.

In what follows, we assume an orientation of the trees in G, with arbitrary root nodes. Fixing the orientation, we let $\mathrm{desc}(i)$ be the set of descendants of i (we assume that $i \in \mathrm{desc}(i)$). For each coalition $S \subseteq N$, we denote by $\mathrm{child}(S)$ the set of *children* of S, namely,

$$\mathrm{child}(S) = \{\, i \in N \setminus S \mid \ i\text{'s parent belongs to } S \,\}.$$

The *height* of a node $i \in N$ is defined inductively as follows:

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

Algorithm 1 An algorithm finding a PAC stable outcome for forest-restricted games

Input: set $\mathcal{S}$ of $m = \lceil \frac{n}{\epsilon} \log \frac{n}{\delta} \rceil$ samples from $\mathcal{D}$
Make a rooted tree with root r by orienting all the edges in E towards the leaves.
Initialize $B_i \leftarrow \emptyset$ and $\pi^{(i)} \leftarrow \emptyset$ for each $i \in N$.
for $t = 0, \ldots, \mathrm{height}(r)$ **do**
  for each node $i \in N$ with $\mathrm{height}(i) = t$ **do**
    set $\mathcal{S}^* = \{\, S \in \mathcal{S} \cap \mathcal{F}_E \mid i \in S \subseteq \mathrm{desc}(i) \ \wedge \ v_j(S) \geq v_j(B_j), \text{ for all } j \in S \setminus \{i\} \,\}$
    choose $B_i \in \mathrm{argmax}\{\, v_i(S) \mid S \in \mathcal{S}^* \cup \{\{i\}\} \,\}$
    set $\pi^{(i)} \leftarrow \{B_i\} \cup \bigcup\{\, \pi^{(j)} \mid j \in \mathrm{child}(B_i) \,\}$
  end for
end for

Given player i, let $\mathcal{B}_i$ be the collection of coalitions B_j for every descendant $j \neq i$ of i, i.e., $\mathcal{B}_i = \{\, B_j \mid j \in \mathrm{desc}(i) \setminus \{i\} \,\}$. For each $i \in N$ and each coalition $X \subseteq N$, we let $P_{\mathcal{B}_i}(X)$ 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 B_j. Now, we define a modified preference order for player i, $\succeq_{\mathcal{B}_i}$, that devalues any coalition X for which $P_{\mathcal{B}_i}(X)$ does not hold.

- If $P_{\mathcal{B}_i}(X)$ and $P_{\mathcal{B}_i}(Y)$, then $X \succ_{\mathcal{B}_i} Y \iff X \succ_i Y$
- If $P_{\mathcal{B}_i}(X)$ but $\neg P_{\mathcal{B}_i}(Y)$, then $X \succ_{\mathcal{B}_i} Y$
- If $\neg P_{\mathcal{B}_i}(X)$, then $\forall Y : Y \succeq_{\mathcal{B}_i} X$

[2]Demange [19] presents the algorithm for non-transferable cooperative utility games on trees where each coalition has a choice of action. A hedonic game is a special case of a non-transferable utility game where each coalition has a unique action.

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

$$\Pr_{S\sim\mathcal{D}}[S \succ_{\mathcal{B}_i} X] \leq \frac{\epsilon}{n}.$$

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

Intuitively, B_i approximates the best coalition i can form with members of the subtree rooted at i. Algorithm 1's objective is to ensure that sampling a coalition S from $\mathcal{D}$ such that $P_{\mathcal{B}_i}(S) \wedge S \succ_i B_i$ is unlikely, namely, the probability of seeing S from $\mathcal{D}$ such that S is better for the highest node i in S than B_i, and every other player in S prefers it to their B_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 $\mathcal{B}_i$ containing B_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-$\frac{\epsilon}{n}$:

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

Note that Inequality (1) is true irrespective of what $\mathcal{B}_i$ 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, B_j is computed based on $\mathcal{S}$, and then B_i is computed based on the same sample. This runs the risk of some underlying dependence between the computation of B_i and the B_j's that is not accounted for in Inequality (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 B_i and maintain complete independence in the samples. However, the smaller sample size used in Algorithm 1 can still provide us with the same guarantee.

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 B_j's for every descendant j of i; finally, based on this, determine the family $\mathcal{B}_i$. Note that regardless of what $\mathcal{B}_i$ 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 1 holds even if $\mathcal{B}_i$ and B_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 1 returns a PAC stable outcome $\pi^{(r)}$. We observe that any coalition included in the returned $\pi^{(r)}$ is a B_i for some i. Note that for every $j \in B_i$, we have that $v_j(B_i) \geq v_j(B_j)$ (line 5). Now, consider any coalition X that strongly blocks $\pi^{(r)}$; let $i = \text{argmax}_{j\in X}\text{height}(j)$. Since X strongly blocks $\pi^{(r)}$,

$$v_j(X) > v_j(\pi^{(r)}(j)) \geq v_j(B_j)$$

for all players $j \in X$. In particular, $v_i(X) > v_i(B_i)$. By construction of B_i and Inequality (1), B_i is top-$\frac{\epsilon}{n}$ for i; that is,

$$\frac{\epsilon}{n} > \Pr_{S\sim\mathcal{D}}[v_i(S) > v_i(B_i)] \geq \Pr_{S\sim\mathcal{D}}[S = X];$$

thus the probability of drawing a coalition such as X from $\mathcal{D}$, i.e. strongly blocking $\pi^{(r)}$ and having $i = \text{argmax}_{j \in X} \text{height}(j)$, 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. (A coalition S weakly blocks a coalition structure if every player weakly prefers S to their current coalition and at least one player in S has a strict preference) We note that in the full information setting, a strict core outcome does not necessarily exist on trees [27].

Remark 4.2 (From Cardinal to Ordinal Preferences)**.** *Note that step 6 of the Algorithm 1 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 1 is players' ranking of coalitions in the sample. In other words, the particular numerical representation of player preferences plays no role. This is a significant departure from the algorithms devised by Sliwinski and Zick [40], where the type of utility representation functions used was crucial for PAC stability.*

Next, we show that Theorem 4.1 is 'tight' in the sense that if the graph G contains a cycle, $\mathcal{H}_G$ is not PAC stabilizable.

Theorem 4.3. *Given a non-forest graph $G = \langle N, E \rangle$, the class $\mathcal{H}_G$ is not PAC stabilizable.*

Proof. Since G is not a forest, there is a cycle in G. Without of loss of generality, let $C = \{1, 2, \ldots, k\}$ be a cycle with $\{i, i+1\} \in E$ for all $i = 1, 2, \ldots, k-1$, and $\{k, 1\} \in E$. Let $S_1 = \{1, 2\}$, $S_2 = \{2, 3\}$, $S_3 = \{3, ..., k, 1\}$. Suppose $\mathcal{D}$ is the uniform distribution on $\{S_1, S_2, S_3\}$ and that the following holds:

$$S_1 \succ_1 S_3, S_2 \succ_2 S_1, S_3 \succ_3 S_2. \tag{2}$$

In this case, nothing beyond (2) can be deduced about the game by examining samples from $\mathcal{D}$. Consider the following games satisfying (2):

1. A game where every player i for $i = 1, 2, 3$ prefers singleton to any other coalition, and any non-singleton coalition is less preferred than S_i, namely,

   $$\{i\} \succ_i S_i \succ_i S'$$

   for any $S' \subseteq N$ such that $i \in S'$ and $S' \neq S_i, \{i\}$.

2. A game where $C = \{1, 2, ..., k\}$ is preferred by $1, ..., k$ to any other coalition, and for $i \in \{1, 2, 3\}$, S_i is preferred to any coalition other than C. The singleton coalition is preferred to any other coalition by all players $j \notin C$.

Both games have a non-empty core. For a partition π to be resistant against deviations supported by $\mathcal{D}$, π has to include $\{1\}$ or $\{2\}$ for the first game, and C for the second game. Hence, it is impossible to achieve $\varepsilon < \frac{1}{3}$ with any confidence $1 - \delta > 0$. □

5 Inferring Tree Interaction Networks from Data

Until now, we assume that the underlying interaction network G was given to us as input; this is, naturally, an assumption that we would like to forgo. Suppose the underlying graph is a forest $T = \langle N, E \rangle$, and consider the question of whether it is possible to infer a forest $T^* = \langle N, E^* \rangle$ that agrees with the original graph with high probability. Let $\mathcal{T}_n$ be the set of all possible trees over n vertices, and let $\mathcal{F}_n$ be the set of all possible forests; $\mathcal{F}_n$ is our hypothesis class for guessing an approximate forest.

By Cayley's formula:

$$|\mathcal{T}_n| = n^{n-2} \tag{3}$$

Any forest can be obtained by choosing a tree, and then removing some k edges from that tree, hence:

$$|\mathcal{F}_n| \leq |\mathcal{T}_n| \sum_{k=0}^{n-1} \binom{n-1}{k} < n^{n-2} n^n \tag{4}$$

We observe the following variant of Theorem 2.1 for finite hypothesis classes.

Theorem 5.1 (Anthony and Bartlett [1])**.** *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*

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

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 $\log |\mathcal{T}_n| < 2n \log n$, 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 labelled 'connected' or 'disconnected' according to T, can we find a forest that is consistent with the labeling? First, we consider an easier question and assume all subsets are connected. The answer to this question is affirmative, and appears in Conitzer et al. [16].

Theorem 5.2 (Conitzer et al. [16])**.** *Let $T = \langle N, E \rangle$ be a tree. 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 T^*.*

Theorem 5.2 pertains to trees, but immediately generalizes to forests by noting that if T is a forest, any tree whose edgeset is a superset of E is a valid solution as well, hence the same algorithm solves the problem.

In other words, if one only observes subsets of feasible coalitions and players' preferences over them, it is possible to find a forest structure consistent with the samples.

Corollary 5.3. *Let $\mathcal{F}_n = \bigcup \{ f_G \mid G$ is a forest over n vertices$\}$ where each $f_G(S) \to \{0, 1\}$ is a function indicating whether S is a connected subgraph of the forest G: $f_G(S) := 1$ if S is connected in G, and 0 otherwise. For a probability distribution $\mathcal{D}$ supporting only connected subgraphs, $\mathcal{F}_n$ is efficiently PAC learnable.*

Corollary 5.3 is immediately implied by (4), Theorems 5.1 and 5.2. Theorem 4.1 assumes that the underlying interaction graph is known to us. Leveraging Corollary 5.3, we now show that this assumption can be forgone; that is, it is possible to PAC stabilize a hedonic game

whose underlying interaction graph is a forest, even if the forest structure is unknown to us. Note that we established that the forest structure can be PAC learned efficiently only if the sample contains exclusively connected coalitions, yet we do not have this requirement for PAC stabilizability.

Theorem 5.4. *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.*

Proof. Suppose ε, δ are given, and there is an unknown forest G, hedonic game $\langle N, \mathcal{V} \rangle \in \mathcal{H}_G$ and a probability distribution $\mathcal{D}$ over coalitions. Let $\mathcal{D}'$ be a distribution obtained from $\mathcal{D}$ by substituting any disconnected coalitions with $\emptyset$. $\mathcal{D}'$ supports only connected coalitions, so by Corollary 5.3, G can be efficiently PAC learned with respect to $\mathcal{D}'$ to obtain G' s.t. $\Pr_{S\sim\mathcal{D}'}[f_{G'}(S) \neq f_G(S)] < \frac{\epsilon}{2}$ with confidence $1 - \frac{\delta}{2}$. Let $\mathcal{D}''$ be a distribution obtained from $\mathcal{D}'$ by substituting any coalitions S s.t. $f_{G'}(S) \neq f_G(S)$ with $\emptyset$. Since $f_{G'}(S) = f_G(S)$ for any S supported by $\mathcal{D}''$, by Theorem 4.1, given G', $\langle N, \mathcal{V} \rangle$ can be PAC stabilized with respect to $\mathcal{D}''$ to obtain a partitioning of the agents π such that $\Pr_{S\sim\mathcal{D}''}[S \text{ strongly blocks } \pi] < \frac{\epsilon}{2}$ with confidence $1 - \frac{\delta}{2}$. For ease of notation, we write $dev(S, \pi)$ whenever S strongly blocks π.

$$
\begin{aligned}
\Pr_{S\sim\mathcal{D}}[dev(S,\pi)] &= \Pr_{S\sim\mathcal{D}}[dev(S,\pi) \wedge S \text{ is connected in } G] \\
&= \Pr_{S\sim\mathcal{D}'}[dev(S,\pi)] \\
&= \Pr_{S\sim\mathcal{D}'}[dev(S,\pi) \wedge f_{G'}(S) = f_G(S)] + \Pr_{S\sim\mathcal{D}'}[dev(S,\pi) \wedge f_{G'}(S) \neq f_G(S)] \\
&\leq \Pr_{S\sim\mathcal{D}'}[dev(S,\pi) \wedge f_{G'}(S) = f_G(S)] + \frac{\varepsilon}{2} \qquad (5) \\
&= \Pr_{S\sim\mathcal{D}''}[dev(S,\pi)] + \frac{\varepsilon}{2} \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon \qquad (6)
\end{aligned}
$$

By construction of G' and π, lines (5) and (6) hold with confidence $1 - \frac{\delta}{2}$ each. We conclude that $\Pr_{S\sim\mathcal{D}}[S \text{ strongly blocks } \pi] \leq \varepsilon$ with confidence at least $1 - \delta$. Since construction of G' and π both require a polynomial number of samples from $\mathcal{D}$, $\mathcal{H}^*$ is efficiently PAC stabilizable. □

Theorem 5.2, 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 the appendix, we study a more general question of whether we can find a tree consistent with both positive (connected coalitions) and negative (disconnected coalitions) examples. As we show in Theorem A.1, introducing the possibility of negative examples makes the problem computationally intractable, even if we restrict ourselves to the hypothesis class of paths. Hence, forests cannot be PAC learned efficiently. It is interesting to note that Theorem 5.4 could be achieved despite this negative result.

6 Conclusions and Future Work

This work establishes a strong connection between interaction structure and the ability to guarantee approximate stability in hedonic games; simply put, we show that if one only knows the underlying interaction structure and nothing more, then one can only obtain PAC stable outcomes if the underlying interaction structure is very well-behaved, i.e. a forest. This result seems to imply a natural tradeoff: our work assumes very little knowledge about underlying player preferences, and thus requires a lot of structure; Sliwinski and Zick [40]

make no assumptions on the underlying interaction network, but assume a more restricted player preference model. It would be interesting to explore 'intermediate' cases; that is, suppose we make some structural assumptions on the interaction network, what classes of player preferences admit PAC stable outcomes?

We make use of tools from computational learning theory in order to analyze hedonic coalition formation. We believe that as a research paradigm, this is a useful and important methodological approach. Hedonic games (and cooperative games in general) have, by and large, seen sparse application. Other game-theoretic methods have been successfully applied by taking a problem-oriented approach (e.g. stable matching for resident-hospital allocation in Chapter 1.1 [29], or Stackelberg games in the security domain [41]); a concrete problem modeled and solved by a hedonic game framework has not yet been identified, to the best of our knowledge; this is despite the wealth of potential application domains, and rich data environments available nowadays (in particular, social network datasets would be particularly agreeable to the type of analysis presented in this work). Our work makes a fundamental connection between data, community structure, and game-theoretic solution concepts; a connection that we hope will result in a more applicable model of strategic collaborative behavior.

References

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

[2] H. Aziz and F. Brandl. Existence of stability in hedonic coalition formation games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 763–770, 2012.

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

[4] M.F. 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.

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

[6] M.F. 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.

[7] M.F. 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.

[8] E. Balkanski, U. Syed, and S. Vassilvitskii. Statistical cost sharing. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 6222–6231, 2017.

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

[10] P. Berman, M. Karpiński, and A. D. Scott. Approximation hardness for short symmetric instances of MAX-3SAT. Technical Report IHES-M-2004-28, Inst. für Informatik, 2004.

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

[12] Kellogg S. Booth and George 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.

[13] 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.

[14] S. Brânzei and K. Larson. Social distance games. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 91–96, 2011.

[15] 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.

[16] 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.

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

[18] V.G. Deineko and G.J. Woeginger. Two hardness results for core stability in hedonic coalition formation games. *Discrete Applied Mathematics*, 161(13):1837–1842, 2013.

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

[20] Michael Dom. Algorithmic aspects of the consecutive-ones property, 2009.

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

[22] 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.

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

[24] 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.

[25] Michel Habib, Ross McConnell, Christophe Paul, and Laurent 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.

[26] 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.

[27] 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.

[28] M.J. Kearns and U.V. Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[29] J. Kleinberg and E. Tardos. *Algorithm Design*. Pearson Education, 2006.

[30] Norbert Korte and Rolf H. Möhring. *A simple linear-time algorithm to recognize interval graphs*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.

[31] Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM Journal on Computing*, 36(2):326–353, 2006.

[32] R. Meir, Y. Zick, E. Elkind, and J.S. 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.

[33] 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.

[34] J. Morgenstern and T. Roughgarden. Learning simple auctions. In *Proceedings of the 29th Conference on Computational Learning Theory (COLT)*, pages 1298–1318, 2016.

[35] R. Myerson. Graphs and cooperation in games. *Mathematics of Operations Research*, 2(3):225–229, 1977.

[36] D. Peters. Graphical hedonic games of bounded treewidth. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pages 586–593, 2016.

[37] 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.

[38] A. Shashua. Introduction to machine learning: Class notes 67577. *arXiv preprint arXiv:0904.3664*, 2009.

[39] 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.

[40] J. Sliwinski and Y. Zick. Learning hedonic games. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2730–2736, 2017.

[41] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[42] G.J. Woeginger. Core stability in hedonic coalition formation. In *Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 33–50, 2013.

Ayumi Igarashi
University of Oxford
Oxford, United Kingdom
Email: ayumi.igarashi@cs.ox.ac.uk

Jakub Sliwinski
ETH Zurich
Zurich, Switzerland
Email: jakvbs@gmail.com

Yair Zick
National University of Singapore
Singapore
Email: dcsyaz@nus.edu.sg

Appendix A The Complexity of Constructing Consistent Paths

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 [3] [12, 30, 17, 23, 25, 31, 26].

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

$$\ell_G(S_j) = \begin{cases} 1 & \text{if } S_j \text{ is connected in } G \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

We say that a graph $G^* = \langle V, E^* \rangle$ is *consistent with* $G = \langle V, E \rangle$ over the samples $\mathcal{S} = \{S_1, \ldots, S_m\} \subseteq V$ if and only if $\ell_{G^*}(S_j) = \ell_G(S_j)$ for all $j \in [m]$. Our objective is to find a path T^* such that $\ell_{T^*}(S_j) = \ell_G(S_j)$ for all j. Theorem A.1 states that it is NP-hard to determine whether such a graph exists.

Theorem A.1. *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 $T^* = \langle N, E \rangle$ 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 [10].
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 $\langle \mathcal{S}, \ell \rangle$ unless the following holds:

[3] 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 [20] for more details).

- 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 $\langle \mathcal{S}, \ell \rangle$.

$\Longrightarrow$: Suppose that there exists a truth assignment $f : X \to \{\mathit{true}, \mathit{false}\}$ 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 1 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 $T^* = \langle N, E \rangle$ consistent with $\langle \mathcal{S}, \ell \rangle$, 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, $\{v_i(2), v_{i+1}(1)\} \in E$ 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 A.2. *For each $i \in [n]$, we have either*

- $\{v_i(1), x_i(1)\}, \{x_i(1), x_i(2)\}, \{x_i(2), v_i(2)\} \in E$*; or*
- $\{v_i(1), \bar{x}_i(1)\}, \{\bar{x}_i(1), \bar{x}_i(2)\}, \{\bar{x}_i(2), v_i(2)\} \in E$.

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.

□

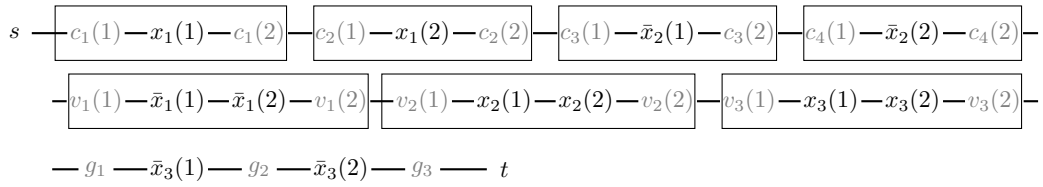

Figure 1: Graph constructed for the formula $\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_3)$ in the proof of Theorem A.1. The formula is satisfied by the mapping f that assigns the opposite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

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. □

To conclude, if one allows observations of both connected and disconnected components, finding a tree consistent with samples is computationally intractable. We note that this does not preclude the existence of efficient heuristics computing consistent tree structures in practice: as previously mentioned, inferring PAC approximations of tree structures has a low communication complexity (Theorem 5.1); thus, given access to strong MILP solvers, we believe that identifying consistent forest structures should be easy in practice.