Inverse Game Theory: Learning Utilities in Succinct Games

Volodymyr Kuleshov[*] Okke Schrijvers[†]

February 12, 2016

Abstract

One of the central questions in game theory deals with predicting the behavior of an agent. Here, we study the inverse of this problem: given the agents' equilibrium behavior, what are possible utilities that motivate this behavior? We consider this problem in arbitrary normal-form games in which the utilities can be represented by a small number of parameters, such as in graphical, congestion, and network design games. In all such settings, we show how to efficiently, i.e. in polynomial time, determine utilities consistent with a given correlated equilibrium. However, inferring both utilities and structural elements (e.g., the graph within a graphical game) is in general NP-hard. From a theoretical perspective our results show that rationalizing an equilibrium is computationally easier than computing it; from a practical perspective a practitioner can use our algorithms to validate behavioral models.

1 Introduction

One of the central and earliest questions in game theory deals with predicting the behavior of an agent. This question has led to the development of a wide range of theories and solution concepts —such as the Nash equilibrium— which determine the players' actions from their utilities. These predictions in turn may be used to inform economic analysis, improve artificial intelligence software, and construct theories of human behavior.

Perhaps equally intriguing is the *inverse* of the above question: given data of the observed behavior of players in a game, how can we infer the utilities that led to this behavior? Surprisingly, this question has received much less attention, even though it arises just as naturally as its more famous converse.

For instance, inferring or *rationalizing* player utilities ought to be an important part of experimental protocols in the social sciences. An experimentalist should test the validity of their model by verifying whether it admits any utilities that are consistent with observed data. More ambitiously, the experimentalist may wish to develop predictive techniques, in which one tries to forecast the agents' behavior from earlier observations, with utilities serving as an intermediary in this process.

Inferring utilities also has numerous engineering applications. In economics, one could design mechanisms that adapt their rules after learning the utilities of their users, in order for instance to

[*]Stanford University, Stanford CA 94305, USA, `kuleshov@stanford.edu`
[†]Stanford University, Stanford CA 94305, USA, `okkes@cs.stanford.edu`

maximize profits. In machine learning, algorithms that infer utilities in a single-agent reinforcement learning setting are key tools for developing helicopter autopilots, and there exists ongoing research on related algorithms in the multi-agent setting.

1.1 Our Contributions.

Previous work on computational considerations for rationalizing equilibria has only considered specific settings, such as matching [15] and network formation games [14]. Here, we instead take a top-down approach and consider the problem in an *arbitrary* normal-form game. Although our results hold generally, the problem becomes especially interesting when the normal-form game is succinct, meaning that player utilities can be represented by a small number of parameters. The number of outcomes in an arbitrary game is in the worst case exponential in the number of players, so even storing the utilities would already require a prohibitive amount of storage. However, many games have additional structure which allows the utilities to be succinctly represented. Indeed, most games studied in the literature —including congestion, graphical, scheduling, and network design games— have this property. Within all succinct games, we establish two main results:

- When the structure of a game (e.g. the graph in a graphical game) is known, we can find utilities that rationalize the equilibrium using a small LP. The LP is polynomial rather than exponential in the number of players and their actions, and hence can be solved in polynomial time using the ellipsoid method. We discuss these results in Section 4.
- If the structure of a succinct game is unknown, inferring both utilities and the correct game structure is NP-hard. We discuss these results in Section 5.

1.2 Related Work

Theoretical Computer Science. Kalyanaraman et al. studied the computational complexity of rationalizing stable matchings [15], and network formation [14]. In the latter case, they showed that game attributes that are local to a player can be rationalized, while other, more global, attributes cannot; this mirrors our observations on the hardness of inferring utilities versus inferring game structure. The forward direction of our problem — computing an equilibrium from utilities — is a central question within algorithmic game theory. Computing Nash equilibria is intractable [9] even for 2 player games [7] (and therefore may be a bad description of human behavior); correlated equilibria, however, are easy to compute in succinct games [20] and can be found using simple iterative dynamics [10, 12]. Our results show that while a Nash equilibrium is hard to compute, it is easy to rationalize. For correlated equilibria, both computing and rationalizing it are feasible.

Economics. Literature on rationalizing agent behavior [22, 2, 23] far predates computational concerns. The field of revealed preference [24] studies an agent who buys different bundles of a good over time, thus revealing more information about its utilities. These are characterized by sets of linear inequalities, which become progressively more restrictive; we adopt this way of characterizing agent utilities in our work as well, but in addition we prove that solving the problem can be done in polynomial time.

Econometrics. Recently, Nekipelov et al. [18] discussed inferring utilities of bidders in online ad auctions, assuming bidders are using a no-regret algorithm for bidding. While no-regret learning agents do converge to a correlated equilibrium, the authors discuss a private-information game, rather than the full information games we consider.

The identification problem in econometrics includes formulations for games, e.g. [6, 17, 5], but their goal is to find a single set of utilities that best describes observed behavior. Since this is often computationally infeasible, much of the literature proposes different estimators. From a theoretical perspective,for the class of succinct games we show that finding *a* set (not necessarily the most likely) of utilities is computationally feasible. Additionally from a practical perspective, we uncover the entire space of valid utilities, which can be used to indicate how confident we should be about assumptions on a model.

Inverse Reinforcement Learning. Algorithms that infer the payoff function of an agent within a Markov decision process [19] are a key tool in building helicopter autopilots [1]. Our work establishes an analogous theory for multi-agent settings. Inverse reinforcement learning has also been used to successfully predict driver behavior in a city [25, 26]; but this work does not learn the utilities of players directly.

Inverse Optimization. Game theory can be interpreted as multi-player optimization, with different agents maximizing their individual objective functions. Recovering the objective function from a solution of a linear program can be solved using a *different* linear program [3]. Our work considers the analogous inverse problem for multiple players and also solves it using an LP.

2 Preliminaries

In a normal-form game $G \triangleq [(A_i)_{i=1}^n, (\mathbf{u}_i)_{i=1}^n]$, a player $i \in \{1, 2, ..., n\}$ has m_i actions $A_i \triangleq \{a_1^i, a_2^i, ..., a_{m_i}^i\}$ and utilities $\mathbf{u}_i \in \mathbb{R}^m$, where $m = \prod_{i=1}^n m_i$ is the cardinality of the joint-action space $A \triangleq \times_{i=1}^n A_i$. An $\mathbf{a} \in A$ is called a joint action of all the players and let $\mathbf{a}_{-i}$ be $\mathbf{a}$ with the action of player i removed. A mixed strategy of player i is a probability distribution $\mathbf{p}_i \in \mathbb{R}^{m_i}$ over the set of actions A_i. A correlated equilibrium (CE) of G is a probability distribution $\mathbf{p} \in \mathbb{R}^m$ over A that satisfies

$$\sum_{\mathbf{a}_{-i}} p(a_j^i, \mathbf{a}_{-i}) u(a_j^i, \mathbf{a}_{-i}) \geq \sum_{\mathbf{a}_{-i}} p(a_j^i, \mathbf{a}_{-i}) u(a_k^i, \mathbf{a}_{-i}) \tag{1}$$

for each player i and each pair of actions a_j^i, a_k^i. This equation captures the idea that no player wants to unilaterally deviate from their equilibrium strategy. Correlated equilibria exist in every game, are easy to compute using a linear program, and arise naturally from the repeated play of learning players [10, 12].

A (mixed) Nash equilibrium is a correlated equilibrium $\mathbf{p}$ that is a product distribution $p(\mathbf{a}) = p_1(a_1) \times ... \times p_n(a_n)$, where the $\mathbf{p}_i \in \mathbb{R}^{m_i}$ are mixed player strategies. In a Nash equilibrium, each player chooses their own strategy (hence the product form), while in a correlated equilibrium the players' actions can be viewed as coming from an outside mediator. A Nash equilibrium exists in every game, but is hard to compute even in the 2-player setting [7].

3 Succinct Games

In general, the dimension m of player i's utility $\mathbf{u}_i$ is exponential in the number of players: if each player has t actions, $\mathbf{u}_i$ specifies a value for each of their t^n possible combinations. Therefore, we restrict our attention to games G that have a special structure which allows the $\mathbf{u}_i$ to be parametrized by a small number of parameters $\mathbf{v}$; such games are called *succinct* [20].

A classical example of a succinct game is a *graphical game*, in which there is a graph H with a node for every player, and the utility of a player depends only on itself and the players on incident nodes in H. Let k be the number of neighbors of i in H, then we only need to specify the utility of i for each combination of actions of $k+1$ players (rather than n). For bounded-degree graphs this greatly reduces the number of parameter value. If the maximum degree in the graph is k and each player has at most t actions, then the total number of utility values per player is at most t^{k+1}, which is independent of n.

Definition 3.1. *A succinct game*

$$G \triangleq [(A_i)_{i=1}^n, (\mathbf{v}_i)_{i=1}^n, (F_i)_{i=1}^n]$$

is a tuple of sets of player actions A_i, parameters $\mathbf{v}_i \in \mathbb{R}^d$, and functions $F_i : \mathbb{R}^d \times A \to \mathbb{R}$ that compute the utility $u_i(\mathbf{a}) = F_i(\mathbf{v}_i, \mathbf{a})$ of a joint action $\mathbf{a}$.

We will further restrict our attention to succinct games in which the F_i have a particular linear form. As we will soon show, almost every succinct game in the literature is also linear. This definition will in turn enable a simple and unified mathematical analysis across all succinct games.

Definition 3.2. *A linear succinct game*

$$G \triangleq [(A_i)_{i=1}^n, (\mathbf{v}_i)_{i=1}^n, (O_i)_{i=1}^n]$$

is a succinct game in which the utilities $\mathbf{u}_i$ are specified by $\mathbf{u}_i = O_i \mathbf{v}_i$, where $O_i \in \{0,1\}^{m \times d}$ is an outcome matrix *mapping parameters into utilities.*

Note that a linear succinct game is a special case of Definition 3.1 with $F_i(\mathbf{v}_i, \mathbf{a}) = (O_i \mathbf{v}_i)_{\mathbf{a}}$, which is the component of $O_i \mathbf{v}$ corresponding to $\mathbf{a}$.

The outcome matrix O_i has an intuitive interpretation. We can think of a set of d distinct outcomes $\mathcal{O}_i$ that can affect the utility of player i. The parameters $\mathbf{v}_i$ specify a utility $\mathbf{v}_i(o)$ for each outcome $o \in \mathcal{O}_i$. When a joint action $\mathbf{a}$ occurs, it results in the realization of a subset $\mathcal{O}_i(\mathbf{a}) \triangleq \{o : (O_i)_{\mathbf{a},o} = 1\}$ of the outcomes, specified by the positions of the non-zero entries of matrix O_i. The utility $u_i(\mathbf{a}) = (O_i \mathbf{v}_i)_{\mathbf{a}}$ equals the sum of valuations of the realized outcomes:

$$u_i(\mathbf{a}) = \sum_{o \in \mathcal{O}_i(\mathbf{a})} v_i(o).$$

Graphical games, which we discussed above, are an example of a succinct game that is linear. In a graphical game with an associated graph H, outcomes correspond to joint actions $\mathbf{a}_{N(i)} = (a^{(k)})_{k \in N(i)}$ by i and its neighbors in H. A joint-action $\mathbf{a}$ activates the single outcome o that is associated to a $\mathbf{a}_{N(i)}$ in which the actions are specified by $\mathbf{a}$. The matrix O_i is defined as

$$(O_i)_{\mathbf{a}, \mathbf{a}_{N(i)}} = \begin{cases} 1 & \text{if } \mathbf{a}, \mathbf{a}_{N(i)} \text{ agree on the actions of } N(i) \\ 0 & \text{otherwise.} \end{cases}$$

3.1 Succinct Representations of Equilibria

Since there is an exponential number of joint actions, a correlated equilibrium $\mathbf{p}$ (which is a distribution over joint actions) may require exponential space to write down. To make sure that the input is polynomial in n, we require that $\mathbf{p}$ be represented as a polynomial mixture of product distributions (PMP) $\mathbf{p} = \sum_{k=1}^{K} \mathbf{q}_k$, where K is polynomial in n, $q_k(\mathbf{a}) = \prod_{i=1}^{n} q_{ik}(a_i)$ and q_{ik} is a distribution over A_i. Correlated equilibria in the form of a PMP exist in every game and can be computed efficiently [20]. A Nash equilibrium is already a product distribution, so it is a PMP with $K = 1$.

3.2 What it Means to Rationalize Equilibria

Finding utilities consistent with an equilibrium $\mathbf{p}$ amounts to finding $\mathbf{u}_i$ that satisfy Equation 1 for each player i and for each pair of actions $a_j^i, a_k^i \in A_i$. It is not hard to show that Equation 1 can be written in matrix form as

$$\mathbf{p}^T C_{ijk} \mathbf{u}_i \geq 0, \tag{2}$$

where C_{ijk} is an $m \times m$ matrix that has the form

$$(C_{ijk})_{(\mathbf{a}_{\text{row}}, \mathbf{a}_{\text{col}})} = \begin{cases} -1 & \text{if } \mathbf{a}_{\text{row}} = (a_j, \mathbf{a}_{-i}^{\text{col}}) \\ 1 & \text{if } \mathbf{a}_{\text{row}} = (a_k, \mathbf{a}_{-i}^{\text{col}}) \\ 0 & \text{otherwise.} \end{cases}$$

This formulation exposes intriguing symmetry between the equilibrium distribution $\mathbf{p}$ and the utilities $\mathbf{u}_i$. By our earlier definitions, the utilities $\mathbf{u}_i$ in a linear succinct game can be written as $\mathbf{u}_i = O_i \mathbf{v}_i$; this allows us to rewrite Equation 2 as

$$\mathbf{p}^T C_{ijk} O_i \mathbf{v}_i \geq 0. \tag{3}$$

While C_{ijk} and O_i are exponentially large in n, their product is not, so in Section 4 we show that we can compute this product efficiently, without constructing C_{ijk} and O_i explicitly.

3.3 Non-Degeneracy Conditions

In general, inferring agent utilities is not a well-defined problem. For instance, Equation 1 is always satisfied by $\mathbf{v}_i = \mathbf{0}$ and remains invariant under scalar multiplication $\alpha \mathbf{v}_i$. To avoid such trivial solutions, we add an additional non-degeneracy condition on the utilities.

Condition 1 (Non-degeneracy)**.** *A non-degenerate vector* $\mathbf{v} \in \mathbb{R}^d$ *satisfies* $\sum_{k=1}^{d} v_k = 1$.

3.4 The Inverse Game Theory Problem

We are now ready to formalize two important inverse game theory problems. In the first problem — Inverse-Utility — we observe L games between n players; the structure of every game is known, but can vary. As a motivating example, consider n drivers that play a congestion game each day over a network of roads and on certain days some roads may be closed. Or consider L scheduling games where different subsets of machines are available on each day. Our goal is to find valuations that rationalize the observed equilibrium of each game.

Definition 3.3 (Inverse-Utility problem). *Given:*

1. *A set of L partially observed succinct n-player games $G_l = [(A_{il})_{i=1}^n, \cdot, (O_{il})_{i=1}^n]$, for $l \in \{1, 2, ..., L\}$.*
2. *A set of L correlated equilibria $(\mathbf{p}_l)_{l=1}^L$.*

Determine succinct utilities $(\mathbf{v}_i)_{i=1}^n$ such that $\mathbf{p}_l$ is a valid correlated equilibrium in each G_l, in the sense that Equation 3 holds for all i and for all $a_j^i, a_k^i \in A_{il}$. Alternatively, report that no such $\mathbf{v}_i$ exist.

In the second problem — Inverse-Game — the players are again playing in L games, but this time both the utilities and the structure of these games are unknown.

Definition 3.4 (Inverse-game problem). *Given:*

1. *A set of L partially observed succinct n-player games $G_l = [(A_{il})_{i=1}^n, \cdot, \cdot]$, for $l \in \{1, 2, ..., L\}$.*
2. *A set of L correlated equilibria $(\mathbf{p}_l)_{l=1}^m$.*
3. *Candidate game structures $(\mathcal{S}_l)_{l=1}^L$, one $\mathcal{S}_l$ per game. Each $\mathcal{S}_l = (S_{lh})_{h=1}^p$ contains p candidate structures. A structure $S_{lh} = (O_{lhi})_{i=1}^n$ specifies an outcome matrix O_{lhi} for each player i.*

Determine succinct utilities $(\mathbf{v}_i)_{i=1}^n$ and a structure $S_l^* = (O_{li}^*)_{i=1}^n \in \mathcal{S}_l$ for each game, such that $\mathbf{p}_l$ is a correlated equilibrium in each $[(A_{il})_{i=1}^n, (\mathbf{v}_i)_{i=1}^n, (O_{li}^*)_{i=1}^n]$, in the sense that

$$\mathbf{p}_l^T C_{ijk} O_{il}^* \mathbf{v}_i \geq 0$$

holds for all i, l and for all $a_j^i, a_k^i \in A_{il}$. Alternatively, report that no such $\mathbf{v}_i$ exist.

An example of this problem is when we observe L graphical games among n players and each game has a different and unknown underlying graph chosen among a set of candidates. We wish to infer both the common $\mathbf{v}$ and the graph of each game.

4 Learning Utilities in Succinct Games

In this section, we show how to solve Inverse-Utility in most succinct games. We start by looking at a general linear succinct game, and derive a simple condition under which Inverse-Utility can be solved. Then we consider specific cases of games (e.g. graphical, congestion, network games), and show 1) that they are succinct and linear, and 2) that they satisfy the previous condition.

4.1 General Linear Succinct Games

To solve Inverse-Utility, we need to find valuations $\mathbf{v}_i$ that satisfy the equilibrium condition (3) for every player i and every pair of actions a_j^i, a_k^i. Notice that if we can compute the product $\mathbf{c}_{ijk}^T \triangleq \mathbf{p}^T C_{ijk} O_i$, then Equation 3 reduces to a simple linear constraint $\mathbf{c}_{ijk}^T \mathbf{v}_i \leq 0$ for $\mathbf{v}_i$. However, the dimensions of C_{ijk} and O_i grow exponentially with n; in order to multiply these objects we must therefore exploit special problem structure. This structure exists in every game for which the following simple condition holds.

Property 1. *Let $A_i(o) = \{\mathbf{a} : (O_i)_{\mathbf{a},o} = 1\}$ be the set of joint-actions that trigger outcome o for player i. The equilibrium summation property holds if*

$$\sum_{\mathbf{a}_{-i}:(a^i_j,\mathbf{a}_{-i})\in A_i(o)} p(\mathbf{a}_{-i}) \tag{4}$$

can be computed in polynomial time for any outcome o, product distribution $\mathbf{p}$, and action a^i_j.[1]

Lemma 4.1. *Let G be a linear succinct game and let $\mathbf{p}$ be a PMP correlated equilibrium. Let $\mathbf{c}^T_{ijk} \triangleq \mathbf{p}^T C_{ijk} O_i$ be the constraint on vector $\mathbf{v}_i$ in Equation 3 for a pair of actions a^i_k, a^i_j. If Property 1 holds, then the components of $\mathbf{c}^T_{ijkj}$ can be computed in polynomial time.*

Proof. For greater clarity, we start with the formulation (1) of constraint (3):

$$\sum_{\mathbf{a}_{-i}} p(a^i_j,\mathbf{a}_{-i})u(a^i_j,\mathbf{a}_{-i}) \geq \sum_{\mathbf{a}_{-i}} p(a^i_j,\mathbf{a}_{-i})u(a^i_k,\mathbf{a}_{-i}) \tag{5}$$

We derive from (5) an expression for each component of $\mathbf{c}_{ijk}$.

Recall that we associate the components of $\mathbf{v}_i$ with a set of outcomes $\mathcal{O}_i$. Let $\mathcal{O}_i(\mathbf{a}) = \{o : O_{(\mathbf{a},o\}} = 1\}$ denote the set of outcomes that are triggered by $\mathbf{a}$; similarly, let $A(o) = \{\mathbf{a} : (O_i)_{\mathbf{a},o} = 1\}$ be the set of joint-actions that trigger an outcome o. The left-hand side of (5) can be rewritten as:

$$\begin{aligned}
\sum_{\mathbf{a}_{-i}} p(a^i_j,\mathbf{a}_{-i})u_i(a^i_j,\mathbf{a}_{-i}) &= \sum_{\mathbf{a}_{-i}} p(a^i_j,\mathbf{a}_{-i}) \sum_{o\in\mathcal{O}_i(a^i_j,\mathbf{a}_{-i})} v_i(o) \\
&= \sum_{o\in\mathcal{O}_i} \sum_{\substack{\mathbf{a}_{-i}:\\(a^i_j,\mathbf{a}_{-i})\in A_i(o)}} p(a^i_j,\mathbf{a}_{-i})v_i(o) \\
&= \sum_{o\in\mathcal{O}_i} v_i(o) \sum_{\substack{\mathbf{a}_{-i}:\\(a^i_j,\mathbf{a}_{-i})\in A_i(o)}} p(a^i_j,\mathbf{a}_{-i})
\end{aligned}$$

Similarly, the right-hand side of (5) can be rewritten as

$$\sum_{\mathbf{a}_{-i}} p(a^i_j,\mathbf{a}_{-i})u_i(a^i_k,\mathbf{a}_{-i}) = \sum_{o\in\mathcal{O}_i} v_i(o) \sum_{\substack{\mathbf{a}_{-i}:\\(a^i_k,\mathbf{a}_{-i})\in A_i(o)}} p(a^i_j,\mathbf{a}_{-i}).$$

Substituting these two expressions into (5) and factoring out $p_i(a^i_j)$ (recall that $\mathbf{p}$ is a product distribution) allows us to rewrite (5) as:

$$\sum_{o\in\mathcal{O}_i} v_i(o) \left[\sum_{\substack{\mathbf{a}_{-i}:\\(a^i_j,\mathbf{a}_{-i})\in A_i(o)}} p(\mathbf{a}_{-i}) - \sum_{\substack{\mathbf{a}_{-i}:\\(a^i_k,\mathbf{a}_{-i})\in A_i(o)}} p(\mathbf{a}_{-i}) \right] \geq 0.$$

[1] Property 1 is closely related to the *polynomial expectation property* (PEP) of [20] which states that the expected utility of a player in a succinct game should be efficiently computable for a product distribution. In fact, the arguments we will use to show that this property holds are inspired by arguments for establishing the PEP.

Notice that the expression in brackets corresponds to the entries of the vector $\mathbf{c}_{ijk}^T$. If $\mathbf{p}$ is a product distribution, then by Property 1, we can compute these terms in polynomial time. If $\mathbf{p}$ is a correlated equilibrium with a PMP representation $\sum_{k=1}^K q_k$, it is easy to see that by linearity of summation we can apply Property 1 K times on each of the terms q_k and sum the results. This establishes the lemma. □ □

Lemma 4.1 suggests solving INVERSE-UTILITY in a game G by means of the following optimization problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} f(\mathbf{v}_i) && (6) \\ \text{subject to} \quad & \mathbf{c}_{ijk}^T \mathbf{v}_i \geq 0 \qquad \forall i, j, k && (7) \\ & \mathbf{1}^T \mathbf{v}_i = 1 \qquad \forall i && (8) \end{aligned}$$

Constraint (7) ensures that $\mathbf{p}$ is a valid equilibrium; by Lemma 4.1, we can compute the components of c_{ijk} if Property 1 holds in G. Constaint (8) ensures that the $\mathbf{v}_i$ are non-degenerate. The objective function (6) selects a set of $\mathbf{v}_i$ out of the polytope of all valid utilities. It is possible to incorporate into this program additional prior knowledge on the form of the utilities or on the coupling of valuations across players.

The objective function f may also incorporate prior knowledge, or it can serve as a regularizer. For instance, we may choose $f(\mathbf{v}_i) = ||\mathbf{v}_i||_1$ to encourage sparsity and make the $\mathbf{v}_i$ more interpretable. We may also use f to avoid degenerate $\mathbf{v}_i$'s; for instance, in graphical games, $\mathbf{c}_{ijk}^T \mathbf{1} = \mathbf{0}$ and constant $\mathbf{v}_i$'s are a valid solution. We may avoid this by adding the $\mathbf{v} \geq 0$ constraint (this is w.l.o.g. when $\mathbf{c}_{ijk}^T \mathbf{1} = \mathbf{0}$) and by choosing $f(\mathbf{v}) = \sum_{o \in \mathcal{O}_i} v(o) \log v(o)$ to maximize entropy.

Note that to simply find a valid $\mathbf{v}_i$, we may set $f(\mathbf{v}_i) = 0$ and find a feasible point via linear programming. Moreover, if we observe L games, we simply combine the constraints $\mathbf{c}_{ijk}$ into one program. Formally, this establishes the main lemma of this section:

Lemma 4.2. *The* INVERSE-GAME *problem can be solved efficiently in any game where Property 1 holds.* □

4.2 Inferring Utilities in Popular Succinct Sames

We now turn our attention to specific families of succinct games which represent the majority of succinct games in the literature [20]. We show that these games are linear and satisfy Property 1; so that, INVERSE-UTILITY can be solved using the optimization problem (6).

Graphical Games. In graphical games [16], a graph H is defined over the set of players; the utility of a player depends only on their actions and those of its neighbors in the graph.

The outcomes for player i are associated to joint-actions $a_{N(i)}$ by the set containing i and its neighbors $N(i)$. A joint-action $\mathbf{a}$ triggers the outcome $a_{N(i)}$ specified by actions of the players in $N(i)$ in $\mathbf{a}$. Formally,

$$(O_i)_{\mathbf{a},\mathbf{a}_{N(i)}} = \begin{cases} 1 & \text{if } \mathbf{a}, \mathbf{a}_{N(i)} \text{ agree on the actions of } N(i) \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to verify that graphical games possess Property 1. Indeed, for any outcome $o = \mathbf{a}_{N(i)}$ and action a^i_j, and letting $a^k_{N(i)}$ be the action of player k in $\mathbf{a}_{N(i)}$, we have

$$\begin{aligned}\sum_{\substack{\mathbf{a}_{-i}: \\ (a^i_j, \mathbf{a}_{-i}) \in A_i(o)}} p(\mathbf{a}_{-i}) &= \prod_{\substack{k \in N(i) \\ k \neq i}} p_k(a^k_{N(i)}) \prod_{\substack{k \notin N(i) \\ k \neq i}} \sum_{\in A_k} p_k(a^k) \\ &= \prod_{\substack{k \in N(i) \\ k \neq i}} p_k(a^k_{N(i)})\end{aligned}$$

Polymatrix Games. In a polymatrix game [13], each player plays i in $(n-1)$ simultaneous 2-player games against each of the other players, and utilities are summed across all these games. Formally, each joint-action triggers $n-1$ different outcomes for player i, one for each pair of actions (a^i, a^j) and thus $u_i(\mathbf{a}) = \sum_{j \neq i} v_i(a^i, a^j)$. The associated outcome matrix is

$$(O_i)_{\mathbf{a},(a^i,a^j)} = \begin{cases} 1 & \text{if } a^i_j \text{ and } a^i_i \text{ are played within } \mathbf{a} \\ 0 & \text{otherwise.} \end{cases}$$

To establish Property 1, observe that when $o = (a^i, a^j)$ is one of the outcomes affecting the utility of player i, we have

$$\sum_{\substack{\mathbf{a}_{-i}: \\ (a^i_j, \mathbf{a}_{-i}) \in A_i(o)}} p(\mathbf{a}_{-i}) = \sum_{\mathbf{a}_{-i} : a^j \in \mathbf{a}_{-i}} p(\mathbf{a}_{-i}) = p_j(a^j).$$

Hypergraphical Games. Hypergraphical games [20] generalize polymatrix games to the case where the simultaneous games involve potentially more than two players. Each instance of a hypergaphical game is associated with a hypergraph H; the vertices of H correspond to players and a hyperedge e indicates that the players connected by e play together in a subgame; the utility of player i is the sum its utilities in all the subgames in which it participates.

The fact that hypergraphical games are linear and possess Property 1 follows easily from our discussion of polymatrix and graphical games.

Congestion Games. In congestion games [21], players compete for a set of resources E (e.g., roads in a city, modeled by edges in a graph); the players' actions correspond to subsets $a^i \subseteq E$ of the resources. After all actions have been played, each player i incurs a cost that equals the sum $\sum_{e \in a^i} d_e(l_e)$ of delays $d_e(\ell_e)$ at each resource e, where $\ell_e(\mathbf{a}) = |\{i : e \in a^i\}|$ denotes the number of players using that resource. In the example involving roads, delays indicate how long it takes to traverse a road based on the congestion.

The outcomes for player i in congestion games are associated with a resource e and the number L of players using that resource; we denote this by $o = (e, L)$. A joint action $\mathbf{a}$ activates the outcomes for the resources in a^i that have $\ell_e(\mathbf{a})$ users. The value $v(o)$ of an outcome $o = (e, L)$ corresponds to the delay experienced on e. Formally, the outcome matrix for a congestion game has the form

$$(O_i)_{\mathbf{a},(e,L)} = \begin{cases} 1 & \text{if } e \in a^i \text{ and } \ell_e(\mathbf{a}) = L \\ 0 & \text{otherwise.} \end{cases}$$

To establish Property 1, we need to show that the expression

$$\sum_{\substack{\mathbf{a}_{-i}: \\ (a^i_j, \mathbf{a}_{-i}) \in A_i(o)}} p(\mathbf{a}_{-i}) = \sum_{\mathbf{a}_{-i}: \ell(\mathbf{a}_{-i}) = L - 1\{e \in a^i_j\}} p(\mathbf{a}_{-i})$$

can be computed for any outcome $o = (e, L)$. Here, $\ell(\mathbf{a}_{-i})$ denotes the number of players other than i using resource e and $1\{e \in a^i_j\}$ equals one if $e \in a^i_j$ and zero otherwise.

The expression $P_L(e) \triangleq \sum_{\mathbf{a}_{-i}: \ell(\mathbf{a}_{-i}) = L} p(\mathbf{a}_{-i})$ can be computed via dynamic programming. Indeed, observe that $P_L(e)$ equals $P[\sum_{j \neq i} B_j(p, e) = L]$, where $B_j(p, e)$ is a Bernoulli random variable whose probability of being one corresponds to the probability $P_{j,e} \triangleq \sum_{a^j : e \in a^j} p_j(a^j)$ of player j selecting an action that includes e. The probabilities $P_{j,e}$ are of course easy to compute. From the $P_{j,e}$ it is easy to compute the $P_L(e)$ using dynamic programming via the recursion:

$$P_L(e) = \sum_{j \neq i} P\left[B_j(p, e) = 1 \cap B_k(p, e) = 0 \ \forall k \neq i, j\right] P_{L-1}(e).$$

Facility Location and Network Design Games. In facility location games [8], players choose one of multiple facility locations, each with a certain cost, and the cost of each facility is then divided by all the players who build it. In network design games [4], players choose paths in a graph to connect their terminals, and the cost of each edge is shared among the players that use it.

These two game types are special cases of congestion games with particular delay functions. These can be handled through additional linear constraints. The earlier discussion for congestion games extends easily to this setting to establish Property 1.

Scheduling Games. In a scheduling game [11, 20], there are M machines and each player i schedules a job on a machine a^i; the job has a machine-dependent running time $t(m, i)$. The player then incurs a cost $t_i(\mathbf{a}) = \sum_{\{j: a^j = a^i\}} t(a^i, j)$ that equals the sum of the running times of all tasks on its machine.

Player outcomes $o = (m, j)$ are associated with a machine m and the task of a player j. The outcome matrix O_i has the form

$$(O_i)_{\mathbf{a},(m,j)} = \begin{cases} 1 & \text{if } m \in a^i \text{ and } m \in a^j \\ 0 & \text{otherwise.} \end{cases}$$

Property 1 can be established by adapting the dynamic programing argument used for congestion games. Note also that congestion games require adding the constraint $v_i(m, k) = v_j(m, k)$ for all i and j in optimization problem (6). We summarize our results in the following theorem.

Theorem 4.3. *The* Inverse-Utility *problem can be solved in polynomial time for the classes of succinct games defined above.* □

5 Learning the Structure of Succinct Games

Unlike the Inverse-Utility problem, for which we have sweeping positive results, the Inverse-game problem is generally hard to solve. We show this under the following non-degeneracy condition for player utilities.

Condition 2 (Non-indifference)**.** *There exist $a^i_j, a^i_k, \mathbf{a}_{-i}$ such that $u_i(a^i_j, \mathbf{a}_{-i}) \neq u_i(a^i_k, \mathbf{a}_{-i})$, where $\mathbf{u}_i = O_i \mathbf{v}_i$.*

Theorem 5.1. *Under Condition 2, it is NP-Hard to solve* Inverse-game *for graphical games, while the corresponding instance of* Inverse-utility *is easy to solve.*

Proof. We reduce from an instance of 3-sat in which for every variable appears in at most $m-2$ clauses.[2] Given an instance of 3-sat with m clauses and n variables, we construct an instance of Inverse-game as follows.

There are $n+1$ players in each game j (for $1 \leq j \leq m$) that are indexed by $i = 0, .., n$. Player 0 has only one action: $a^{(0)}$. Every other player $i \geq 1$ has 2 actions: $a^{(i)}_T$ and $a^{(i)}_F$.

Every game j is associated with a clause C_j. Game j has an unknown underlying graph that is chosen in the set of graphs $\mathcal{S}_j = \{H_{j1}, H_{j2}, H_{j3}\}$, where H_{jk} is the graph consisting of only a single edge between player 0 and the player associated with the variable that appears as the k-th literal in clause j. In other words, in each game, only one of three possible players is connected to player 0 by an edge.

The utilities v_i of each player $i \geq 1$ are four-dimensional: they specify two values $v_i(a^{(i)}_T), v_i(a^{(i)}_F)$ when player i is not connected by an edge to player 0, and two values $v_i(a^{(i)}_T; a^{(0)}), v_i(a^{(i)}_F; a^{(0)})$ when they are.

For every clause C_j, we also define an input equilibrium p_j. Each p_j is a pure strategy Nash equilibrium and decomposes into a product $p_j = \prod_{i=1}^n p_{ji}$. Since player 0 has only one action, p_{j0} is defined trivially. When variable x_i appears in clause C_j, we define the probability of player $i \geq 1$ playing action $a^{(i)}_T$ as

$$p_{ji}\left(a^{(i)} = a^{(i)}_T\right) = \begin{cases} 1 & \text{if } x_i \text{ is positively in clause } C_j \\ 0 & \text{if } x_i \text{ is negated in clause } C_j, \end{cases}$$

and $p_{ji}(a^{(i)} = a^{(i)}_F) = 1 - p_{ji}(a^{(i)} = a^{(i)}_T)$. When variable x_i does not appear in clause C_j, we set the strategy in one such game j (chosen arbitrarily) to be $p_{ji}(a^{(i)} = a^{(i)}_T) = 1$, and in the remaining games we set $p_{ji}(a^{(i)} = a^{(i)}_F) = 1$.

This completes the construction of the game. Next, we will introduce some notation and make a few observations, before showing that 3-sat is encoded in the constructed game.

Let $\Delta_i(a^{(i)}_T \to a^{(i)}_F) \triangleq v_i(a^{(i)}_F, a^{(0)}) - v_i(a^{(i)}_T, a^{(0)})$ be the gain to player $i \geq 1$ for deviating from $a^{(i)}_T$ to $a^{(i)}_F$ when they are not connected to player 0, and let $\Delta_i(a^{(i)}_T \to a^{(i)}_F; a^{(0)}) \triangleq v_i(a^{(i)}_F, a^{(0)}; a^{(0)}) - v_i(a^{(i)}_T, a^{(0)}; a^{(0)})$ be the gain when they are.

Observe that the constraint of player $i \geq 1$ when they are not connected to player 0 is of the form

$$p_{ij}(a^{(i)}_F)\Delta_i(a^{(i)}_T \to a^{(i)}_F) + p_{ij}(a^{(i)}_T)\Delta_i(a^{(i)}_F \to a^{(i)}_T) \geq 0,$$

and the constraint when i and 0 are connected is similar.

Because each variable x_i appear in at most $m-2$ clauses, and because of how we defined the probabilities, the following constraints act on $\Delta_i(a^{(i)}_T \to a^{(i)}_F)$: in one game j_1 such that x_i is not in C_j we have $\Delta_i(a^{(i)}_T \to a^{(i)}_F) \geq 0$, and in all other such games we have $\Delta_i(a^{(i)}_T \to a^{(i)}_F) \leq 0$. Because by

[2]This is without loss of generality: if there is a variable that appears in all clauses, we can construct two 2-sat instances which can be solve efficiently, if a variable appears in all but one clause we can duplicate that clause.

definition $\Delta_i(a_T^{(i)} \to a_F^{(i)}) = -\Delta_i(a_F^{(i)} \to a_T^{(i)})$, we must have $\Delta_i(a_T^{(i)} \to a_F^{(i)}) = \Delta_i(a_F^{(i)} \to a_T^{(i)}) = 0$. Because of non-degeneracy constraints on the utilities v, this implies that $\Delta_i(a_F^{(i)} \to a_T^{(i)}; a^{(0)}) \neq 0$. This concludes the observations.

We now show how 3-SAT is encoded in the game we defined. Suppose that we have an valid assignment of utilities and a structure; this leads to a satisfying assignment in 3-SAT: we simply set to be true in clause j the literal associated with the player that is connected to player 0 in game j.

Clearly there will be one true literal per clause. We only need to show that both the literal and its negation are never chosen. Suppose that was the case and there were two clauses j_1, j_2 such that x_i is chosen in j_1 and $\bar{x}_i$ is chosen in j_2. Then, player i's constraint in j_1 is $\Delta_i(a_T^{(i)} \to a_F^{(i)}; a^{(0)}) > 0$ and in j_2 it is $\Delta_i(a_F^{(i)} \to a_T^{(i)}; a^{(0)}) = -\Delta_i(a_T^{(i)} \to a_F^{(i)}; a^{(0)}) > 0$ and there cannot be utilities that satisfy both these constraints.

Finally we show that a satisfying 3-SAT assignment leads to valid utilities in INVERSE-GAME. First, set all utilities $v_i(a_T^{(i)}), v_i(a_F^{(i)})$ to zero. Set the utility of player 0 to one. Pick a true literal in each clause j and connect the corresponding player in game j to player 0. We have to show that we can find valid utilities for all players. Clearly, that is feasible for player 0. We claim that it is also possible for any player $i \geq 1$. First, set $\Delta_i(a_T^{(i)} \to a_F^{(i)}) = -\Delta_i(a_F^{(i)} \to a_T^{(i)}) = 0$. Next, notice if x_i is true, then all constraints involving player i are $\Delta_i(a_T^{(i)} \to a_F^{(i)}; a^{(0)}) > 0$), and if x_i is false, all constraints involving player i are $\Delta_i(a_F^{(i)} \to a_T^{(i)}; a^{(0)}) > 0$. In both cases we can find valid utilities to satisfy these constraints, and so INVERSE-GAME can be solved.

Finally, observe that the corresponding instance of INVERSE-UTILITY (i.e., the one in which the correct graph is pre-specified in advance) is easy to solve. The number of players and actions is very small. Furthermore, it is easy to enforce Condition 2, as there are only two actions for each player, and two values of a_{-i}^i to consider; the condition can thus be enforced by adding a small (polynomial) number of constraints to the problem. □

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–, New York, NY, USA, 2004. ACM.

[2] S. N. Afriat. The construction of utility functions from expenditure data. *International Economic Review*, 8(1):pp. 67–77, 1967.

[3] Ravindra K. Ahuja and James B. Orlin. Inverse optimization. *Operations Research*, 49(5):771–783, 2001.

[4] E. Anshelevich, A. Dasgupta, J. Kleinberg, . Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

[5] Patrick Bajari, Han Hong, and Stephen P Ryan. Identification and estimation of a discrete game of complete information. *Econometrica*, 78(5):1529–1568, 2010.

[6] Timothy F Bresnahan and Peter C Reiss. Empirical models of discrete games. *Journal of Econometrics*, 48(1):57–81, 1991.

[7] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)*, 56(3):14, 2009.

[8] Byung-Gon Chun, Kamalika Chaudhuri, Hoeteck Wee, Marco Barreno, Christos H. Papadimitriou, and John Kubiatowicz. Selfish caching in distributed systems: A game-theoretic analysis. In *Proceedings of the 23rd Annual ACM Symposium on PODC*, PODC '04, pages 21–30, New York, NY, USA, 2004. ACM.

[9] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 71–78, New York, NY, USA, 2006. ACM.

[10] Dean P. Foster and Rakesh V. Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21(12):40 – 55, 1997.

[11] Dimitris Fotakis, Spyros Kontogiannis, Elias Koutsoupias, Marios Mavronicolas, and Paul Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 123–134. Springer Berlin Heidelberg, 2002.

[12] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[13] Joseph T Howson Jr. Equilibria of polymatrix games. *Management Science*, 18(5-part-1):312–318, 1972.

[14] S. Kalyanaraman and C. Umans. The complexity of rationalizing network formation. In *Foundations of Computer Science, 2009. FOCS '09. 50th Annual IEEE Symposium on*, pages 485–494, Oct 2009.

[15] Shankar Kalyanaraman and Christopher Umans. The complexity of rationalizing matchings. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation*, volume 5369 of *Lecture Notes in Computer Science*, pages 171–182. Springer Berlin Heidelberg, 2008.

[16] Michael Kearns, Michael L. Littman, and Satinder Singh. Graphical models for game theory. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pages 253–260, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[17] Wietze Lise. Estimating a game theoretic model. *Computational Economics*, 18(2):141–157, 2001.

[18] Denis Nekipelov, Vasilis Syrgkanis, and Eva Tardos. Econometrics for learning agents. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 1–18, New York, NY, USA, 2015. ACM.

[19] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

[20] Christos H. Papadimitriou and Tim Roughgarden. Computing correlated equilibria in multi-player games. *J. ACM*, 55(3):14:1–14:29, August 2008.

[21] RobertW. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[22] Paul A. Samuelson. Consumption theory in terms of revealed preference. *Economica*, 15(60):pp. 243–253, 1948.

[23] Hal R. Varian. The nonparametric approach to demand analysis. *Econometrica*, 50(4):pp. 945–973, 1982.

[24] Hal R Varian. Revealed preference. *Samuelsonian Economics and the Twenty-First Century*, 2006.

[25] Kevin Waugh, Brian D. Ziebart, and J. Andrew Bagnell. Computational rationalization: The inverse equilibrium problem. 2013.

[26] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proc. Ubicomp*, pages 322–331, 2008.