Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Resource Graph Games: A Compact Representation for Games with Structured Strategy Spaces

Albert Xin Jiang,[1] **Hau Chan,**[1] **Kevin Leyton-Brown**[2]
[1]Department of Computer Science, Trinity University, San Antonio, TX 78212, USA
{xjiang, hchan}@trinity.edu
[2]Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1Z4, Canada
kevinlb@cs.ubc.ca

Abstract

In many real-world systems, strategic agents' decisions can be understood as complex—i.e., consisting of multiple sub-decisions—and hence can give rise to an exponential number of pure strategies. Examples include network congestion games, simultaneous auctions, and security games. However, agents' sets of strategies are often structured, allowing them to be represented compactly. There currently exists no general modeling language that captures a wide range of commonly seen strategy structure and utility structure.

We propose Resource Graph Games (RGGs), the first general compact representation for games with structured strategy spaces, which is able to represent a wide range of games studied in literature. We leverage recent results about multilinearity, a key property of games that allows us to represent the mixed strategies compactly, and, as a result, to compute various equilibrium concepts efficiently. While not all RGGs are multilinear, we provide a general method of converting RGGs to those that are multilinear, and identify subclasses of RGGs whose converted version allow efficient computation.

1 Introduction

There has been increasing interest in using game theory to model real-world multiagent systems, and in the computation of game-theoretic solution concepts given such a model. For games with large numbers of agents and actions, the standard normal form game representation requires exponential space and is thus not a practical option as a basis for computation. Fortunately, most large games of practical interest have highly structured *utility functions*, and thus it is possible to represent them compactly. This helps to explain why people are able to reason about these games in the first place: we understand the payoffs in terms of simple relationships rather than in terms of enormous lookup tables. A line of research thus exists to look for *compact game representations* that are able to succinctly describe structured games, including work on graphical games (Kearns, Littman, and Singh 2001), multi-agent influence diagrams (Koller and Milch 2001) and action-graph games (Jiang, Leyton-Brown, and Bhat 2011); as well as work on efficient algorithms for computing solution concepts such as Nash equilibrium (Ortiz and Kearns 2003; Daskalakis, Fabrikant, and Papadimitriou 2006) and (coarse) correlated equilibrium (Kakade et al. 2003; Papadimitriou and Roughgarden 2008) given compactly represented games.

In many real-world domains, each player needs to make a decision that consists of multiple sub-decisions (e.g., assigning a set of resources, ranking a set of options, or finding a path in a network), and hence can have an exponential number of pure strategies. However, this space of pure strategies is often *structured*, meaning that it can be represented compactly. To compactly represent games with *structured strategy spaces*, we need compact ways to represent both the utility functions and the strategy spaces themselves.

Several classes of multi-player game models studied in the recent literature have structured strategy spaces, including network congestion games (Daskalakis, Fabrikant, and Papadimitriou 2006), simultaneous auctions and other multi-item auctions (Vorobeychik 2008; Rabinovich et al. 2009), dueling algorithms (Immorlica et al. 2011), integer programming games (Köppe, Ryan, and Queyranne 2011), Blotto games (Ahmadinejad et al. 2016), and security games (Korzhyk, Conitzer, and Parr 2010; Tambe 2011). These authors proposed compact game representations that are suitable for their specific domains and computation needs. However, there exists no general modeling language that captures a wide range of commonly seen strategy and utility structure. As a result, given a new domain with structured strategy spaces that does not fall into the existing models, it is not obvious how to represent the game and compute its equilibria. Existing general-purpose compact representations, including graphical games and action graph games, represent each player's set of pure strategies explicitly, which implies that games with structured strategy spaces require exponential space in these representations.

In this paper, we propose *Resource Graph Games* (RGGs), a compact representation for games with structured strategy spaces. RGGs generalize action graph games and congestion games; at a high level, an RGG consists of a graphical representation of utility functions together with a general representation of strategy spaces as convex polytopes. We show that RGGs can compactly encode a wide range of games studied in the literature, as well as new classes of games for which existing methods do not apply.

We also provide efficient algorithms for computing solution concepts given RGGs. To this end, we make use of

a recent result (Chan et al. 2016): if a game is *multilinear* (the utility functions are linear in each player's strategy), and furthermore if there exist polynomial-time algorithms for two subproblems UtilityGradient and PolytopeSolve, then (i) mixed strategies can be represented compactly, (ii) a coarse correlated equilibrium (CCE) can be computed in polynomial time, and (iii) the problem of finding a Nash equilibrium is in PPAD, which implies that the problem can be polynomially reduced to the problem of finding a Nash equilibrium in a polynomial-sized bimatrix game.

We show that all RGGs can be formulated as multilinear games, after adding a polynomial number of auxiliary resource nodes. Furthermore, UtilityGradient can be computed in polynomial time. However, PolytopeSolve may require exponential time. We identify subclasses of RGGs for which such polynomial-time subroutines can be constructed. Thus, we are able to compute various solution concepts for those games efficiently. Finally, we identify a natural subclass of RGGs that are multilinear (without any reformulation): those in which a player's sub-decisions do not directly influence each other's utilities.

A related line of work aims to compactly represent games' decisions and utilities using either Boolean logic (Harrenstein et al. 2001; Dunne and Wooldridge 2012) or constraint satisfaction problems (Nguyen, Lallouet, and Bordeaux 2013; Nguyen and Lallouet 2014). Each player in these formulations can control multiple decision variables, and as a result these formulations can model games with structured strategy spaces. These works focus on computing pure-strategy Nash equilibrium (PSNE) and its variants, using techniques from Boolean satisfiability and constraint programming respectively. Our work focuses on computation of mixed-strategy based solution concepts including Nash equilibrium and coarse correlated equilibrium, which always exist in finite games, unlike PSNE.

2 Preliminaries

A game is specified by (N, S, u), where $N = \{1, \ldots, n\}$ is the set of players. Each player $i \in N$ chooses from a finite set of pure strategies S_i. Denote by $s_i \in S_i$ a pure strategy of i. Then $S = \prod_i S_i$ is the set of pure-strategy profiles. Moreover, $u = (u_1, \ldots, u_n)$ are the utility functions of the players, where the utility function of player i is $u_i : S \to \mathbb{R}$.

The *normal form* explicitly represents the strategy set S_i and the utility function u_i for each player i. Thus, the size of the representation is on the order of $n|S| = n\prod_i |S_i|$.

A mixed strategy σ_i of player i is a probability distribution over her pure strategies. Let $\Sigma_i = \Delta(S_i)$ be i's set of mixed strategies, where $\Delta(\cdot)$ denotes the set of probability distributions over a finite set. Denote by $\sigma = (\sigma_1, \ldots, \sigma_n)$ a mixed strategy profile, and $\Sigma = \prod_i \Sigma_i$ the set of mixed strategy profiles. Denote by σ_{-i} the mixed strategy profile of players other than i. σ induces a probability distribution over pure strategy profiles. Denote by $u_i(\sigma)$ the expected utility of player i under σ: $u_i(\sigma) = E_{s\sim\sigma}[u_i(s)] = \sum_{s\in S} u_i(s)\prod_{k\in N}\sigma_k(s_k)$, where $\sigma_k(s_k)$ is player k's probability of playing the pure strategy s_k.

Player i's strategy σ_i is a best response to σ_{-i} if $\sigma_i \in \arg\max_{\sigma_i'\in\Sigma_i} u_i(\sigma_i', \sigma_{-i})$. A mixed strategy profile σ is a Nash equilibrium (NE) if for each player $i \in N$, σ_i is a best response to σ_{-i}.

Denote by $\mathbb{Z}_+$ the set of nonnegative integers. A *rational polytope* is defined by a set of inequalities with integer coefficients; formally $P = \{x \in \mathbb{R}^m | Dx \le f\}$ is a rational polytope if D and f consist of integers.

Polytopal Strategy Spaces We are interested in games in which a pure strategy has multiple components. Without loss of generality, if each pure strategy of player i has m_i components, we can associate each such pure strategy with an m_i-dimensional nonnegative integer vector. Then the set of pure strategies for each player i is $S_i \subset \mathbb{Z}_+^{m_i}$. In general the number of integer points in S_i can grow exponentially in m_i. Thus, we need a compact representation of S_i.

In most studies of games with structured strategy spaces, each S_i can be expressed as the set of integer points in a rational polytope P_i, i.e., $S_i = P_i \cap \mathbb{Z}_+^{m_i}$. Then, in order to represent the strategy space, we only need to specify the set of linear constraints defining P_i, with each linear constraint requiring us to store $O(m_i)$ integers. We call a game with this property *a game with polytopal strategy spaces*.

Multilinear Games When $|S_i|$ is exponential, representing a mixed strategy σ_i explicitly would take exponential space. (Chan et al. 2016) showed that mixed strategies can be compactly represented if the game is *multilinear*.

Definition 1. *Consider a game Γ with polytopal strategy sets $S_i = P_i \cap \mathbb{Z}_+^{m_i}$ for each i. Γ is a multilinear game if*

1. *for each player i, $\exists U^i \in \mathbb{R}^{\prod_{k\in N} m_k}$ such that $\forall s \in S$,*

$$u_i(s) = \sum_{(j_1\ldots j_n)\in\prod_k[m_k]} U^i_{j_1\ldots j_n} \prod_{k\in N} s_{k,j_k},$$

   where $[m_k] = \{1, \ldots, m_k\}$; in particular, given a fixed s_{-j}, u_i is a linear function of s_j.
2. *The extreme points (i.e. vertices) of P_i are integer vectors, which implies that $P_i = conv(S_i)$, the convex hull of S_i.*

Given a mixed strategy σ_i, the *marginal vector* corresponding to σ_i is the expectation over the pure strategy space S_i induced by the distribution $\sigma_i : \pi_i = E_{\sigma_i}[s_i] = \sum_{s_i\in S_i} \sigma_i(s_i)s_i$. The set of marginal vectors is exactly $conv(S_i) = P_i$. (Chan et al. 2016) showed that given a multilinear game, $u_i(\sigma) = \sum_{(j_1\ldots j_n)\in\prod_k[m_k]} U^i_{j_1\ldots j_n}\prod_{k\in N}\pi_{k,j_k}$. That is, marginal vectors capture all payoff-relevant information about mixed strategies, and thus we can use them to compactly represent the space of mixed strategies.

Given the *marginal strategy profile* $\pi = \{\pi_i\}_{i\in N}$, we slightly abuse notation and denote by $u_i(\pi)$ player i's expected utility under π. Due to multilinearity, after fixing the strategies of players $N \setminus \{k\}$, $u_i(\pi)$ is a linear function of π_k. We define the *utility gradient* of player i with respect to player k's marginal, $\nabla_k(u_i(\pi_{-k})) \in \mathbb{R}^{m_k}$, to be the vector of coefficients of this linear function. Formally, $\forall j_k \in [m_k]$, $(\nabla_k u_i(\pi_{-i}))_{j_k} \equiv \sum_{(j_1,..,j_{k-1},j_{k+1},..,j_n)\in\prod_{\ell=1}^{N\setminus\{k\}}[m_\ell]} U^i_{j_1\ldots j_n}\prod_{\ell\in N\setminus\{k\}}\pi_{\ell,j_\ell}$.

Equilibrium Computation in Multilinear Games (Chan et al. 2016) identify two problems as the key subtasks for equilibrium computation: First, $(UtilGradient)$ given a compactly represented game that satisfies multilinearity, given players $i, k \in N$, and π_{-k}, compute $\nabla_k(u_i(\pi_{-k}))$. Second, $(PolytopeSolve)$ given a compactly represented game with polytopal strategy space, player i, and a vector $d \in \mathbb{R}^{m_i}$, compute $\arg\max_{x \in P_i} d^T x$.

They note that if PolytopeSolve can be solved in polynomial time, then there is a polynomial time procedure to generate mixed strategies that are consistent with the given marginals. Moreover, they show that given a multilinear game, if there are polynomial-time procedures for UtilGradient and PolytopeSolve, then best response against marginals can be computed in polynomial time, an exact CCE can be found in polynomial time, and computing a Nash equilibrium is in PPAD.

3 Resource Graph Games

Informally, an RGG is played on a set $\mathcal{A}$ of *resources*, with each player able to choose a subset of the resources. Thus, each pure strategy is a $|\mathcal{A}|$-dimensional 0–1 vector, and we represent the set of pure strategies as a polytopal strategy space. There is a resource graph, whose nodes are resources and whose directed edges induce a neighborhood for each resource. Each player i's utility is the sum of contributions from each of his chosen resources. The utility contribution from choosing a resource $\alpha \in \mathcal{A}$ is a function of the configuration (vector of counts of the number of players choosing each resource) of α's neighborhood.

Formally, a Resource Graph Game (RGG) is specified by the tuple $(N, \mathcal{A}, \{S_i\}, G, \{u^\alpha\})$, where:

- $N = \{1, \ldots, n\}$ is the set of players.
- $\mathcal{A}$ is a set of resources.
- Each pure strategy $s_i \in S_i$ of player i corresponds to a subset of resources, represented by a $|\mathcal{A}|$-dimensional 0–1 indicator vector.
- S_i is represented as a polytopal strategy space: $S_i = P_i \cap \{0,1\}^{|\mathcal{A}|}$ where $P_i = \{x \in [0,1]^{|\mathcal{A}|} | D_i x \leq f_i\}$, $D_i \in Z^{\ell_i \times |\mathcal{A}|}$ and $f_i \in Z^{\ell_i}$, i.e., P_i is a rational polytope defined by ℓ_i linear constraints. Let $s_{i\alpha}$ be the component corresponding to resource node $\alpha \in \mathcal{A}$.
- Given a pure strategy profile $s = (s_1 \ldots s_n)$, the configuration $c \in \mathbb{Z}_+^{|\mathcal{A}|}$ is a vector of integers representing the total number of agents who have selected each resource:[1] $c = \sum_{i \in N} s_i$.
- The resource graph $G = (\mathcal{A}, E)$ is a directed graph, where self-edges are allowed. The neighborhood of α, denoted by $\nu(\alpha)$, is the set of nodes with edges going into α. Denote by $c^{(\alpha)} \in \mathbb{Z}_+^{\nu(\alpha)}$ the configuration over the neighborhood of α. Let $C^{(\alpha)} \subset \mathbb{Z}_+^{\nu(\alpha)}$ be the set of possible configurations over the neighborhood of α.
- For each resource node α, define utility function $u^\alpha : C^{(\alpha)} \to \mathbb{R}$. $u^\alpha(c^{(\alpha)})$ represents the utility contribution of using resource α, when the pure strategy profile induces a configuration $c^{(\alpha)}$ over the neighborhood of α.
- The utility of player i is then aggregated[2] over the resources chosen by s_i:

$$u_i(s) = \sum_{\alpha:\, s_{i\alpha}=1} u^\alpha(c^{(\alpha)}) = \sum_{\alpha \in \mathcal{A}} s_{i\alpha} u^\alpha(c^{(\alpha)}). \quad (1)$$

3.1 Properties of RGGs

What is the size of the RGG representation? In general, the representation size grows polynomially as long as the graph has bounded in-degree.

Proposition 1. *An RGG representation stores $O(|\mathcal{A}|(n^{\mathcal{I}} + \sum_{i=1}^{n} \ell_i))$ numbers, where $\mathcal{I}$ is the maximum in-degree of the resource graph.*

Since each $S_i \subset \{0,1\}^{|\mathcal{A}|}$, RGGs can represent games with as many as $2^{|\mathcal{A}|}$ pure strategies per player.

Another natural question is the generality of RGGs. One observation is that the pure strategies in RGGs are limited to vertices of the 0–1 hypercube $\{0,1\}^{|\mathcal{A}|}$. What if we want to represent pure strategies in Z^{m_i} that involve larger integers? For example, in the context of resources, a player i could decide to use different integer amounts $j \in \{0, 1, \ldots, J\}$ of resource α. We could extend our definition of RGGs to allow this, but, in this paper, we restrict to 0–1 vectors for notational simplicity. Furthermore, it turns out that we can reduce the general integer vector case to the case of 0–1 vectors by introducing additional resource nodes. More specifically, we create resource nodes $\alpha^0, \ldots, \alpha^J$, and allow player i to choose exactly one node α^j from this set, indicating that j units of resource are used. This can be enforced by the linear constraint $\sum_{j=0}^{J} s_{i\alpha^j} = 1$. The usage amount by i on the original resource can be expressed as $\sum_{j=0}^{J} j s_{i\alpha^j}$, which can be substituted into any constraints on this usage amount, resulting in a set of linear constraints on the 0–1 vector s_i.

Another potential limitation is that each player only receives utility contributions from the resources that he choses. What if we want to model cases where player i also receives utility from not choosing a resource α? For example, in a security scenario, by not protecting a target the defender may incur costs if that target is attacked. We can model this using RGGs by creating an additional resource node α^0, and adding constraints so that $s_{i\alpha^0} = 1$ whenever $s_{i\alpha} = 0$ and vice versa. Then we can define a utility function on α^0, which will be triggered whenever α is not chosen. In Example 3 below we illustrate this technique.

Indeed, the fact that we could encode this scenario was not a coincidence: RGGs can represent arbitrary games. Since

[1] Results in this paper can be adapted to the more general case where configuration on resource α is defined to be $c(\alpha) = *_{i \in N} g_i(s_{i\alpha})$ where $*$ is a binary operator and $g_i : \{0,1\} \to \mathbb{Z}$. This is analogous to contribution-independent function nodes in action-graph games (Jiang, Leyton-Brown, and Bhat 2011).

[2] For certain games, it might make sense to use another aggregation operator instead of summation, such as $\min$ and $\max$. We do not further discuss such games in this paper.

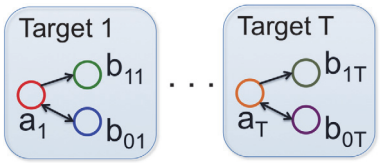

Figure 1: RGG representation of a T-target security game.

AGGs are a special case of RGGs and AGGs can represent arbitrary normal form games (Jiang, Leyton-Brown, and Bhat 2011), we have the following.

Proposition 2 (Generality of RGGs)**.** *Any finite game can be represented as an RGG.*

The following examples show that RGGs can *compactly* represent a wide range of games studied in the literature.

Example 1 (Congestion Games)**.** *Each resource in the congestion game corresponds to a resource node in the RGG representation. The resource graph contains only self-edges. Each pure strategy is a subset of resources; in the RGG this is represented as an indicator vector s_i for the subset. P_i is then the convex hull of the set of pure strategies S_i. u^α corresponds to the congestion cost function for resource α.*

Example 2 (Network Congestion Games)**.** *In network congestion games, each player i has a fixed source node and sink node and needs to choose a path from source to sink. Player i's total cost is the sum of costs incurred at each edge along the chosen path. In the RGG representation, resource nodes correspond to the edges of the routing network. S_i is the set of 0-1 vectors of resources corresponding to paths from source node to sink node. The polytope P_i can be represented compactly with a set of flow constraints, as in (Daskalakis, Fabrikant, and Papadimitriou 2006). Moreover, the resource graph consists of only self-edges and $\{u^\alpha\}$ corresponds to the congestion cost functions.*

Example 3 (Security Games)**.** *Consider a security game (Tambe 2011) between a defender and an attacker. The defender can only protect m targets and attacker can attack one target in T. The defender and attacker's utilities depend on whether the attacked target is covered.*

We can represent security games as RGGs. For each target $t \in T$, we have a resource node a_t for the attacker and two resource nodes b_{0t} and b_{1t} for the defender. Let $\mathcal{A} = \bigcup_{t\in T}\{a_t, b_{0t}, b_{1t}\}$ be the set of resource nodes. Then, the strategy set of the attacker is $S_a = \{x \in \{0,1\}^{3|T|} \mid \sum_{a_t} x_{at} = 1, x_{b0t} = x_{b1t} = 0 \forall t \in T\}$. For the defender, we construct a constraint matrix D_b of size $|T|+1$ by $3|T|$. In particular, the first row of D_b is $d_b = (d_{b\alpha})_{\alpha\in\mathcal{A}}$ where $d_{b\alpha} = 1$ at $\alpha = b_{1t}$ for every $t \in T$; and then for each $t \in T$, there is a row of $d_t = (d_{t\alpha})_{\alpha\in A}$ such that $d_{t\alpha} = 1$ only at b_{0t} and b_{1t}. Thus, the strategy set of a defender is $S_b = \{x \in \{0,1\}^{3|T|} \mid D_b x = (m, 1, ..., 1)^T\}$. In other words, the constraint matrix ensures that the defender can protect m targets and can either protect a target or not protect a target (but never both). Figure 1 illustrates the resource graph. This graph is constructed so that for every $t \in T$, there is an edge from b_{1t} to a_t, an edge from a_t to b_{1t}, and an edge from a_t to b_{0t}. The utilities of u^{a_t}, $u^{b_{1t}}$, and $u^{b_{0t}}$, can be defined appropriately for each target $t \in T$.

Not only can RGGs be used to represent existing compact classes of games, they can also be used to compactly model new instances of game-theoretic problems.

Example 4 (Local-Network Congestion Games)**.** *Using RGGs, we can study local-network congestion games in which the congestion of an (outgoing) edge affects the congestion of the other adjacent (incoming) edges (i.e., the adjacent edges that have the same destination as the source of the outgoing edge). This game class is inspired by the real-life situation of traffic congestion, where extreme congestion in one road can cause delays on adjacent roads.*

As in Example 2, resource nodes correspond to the edges and each player i can play the actions in S_i. The resource graph is no longer just "self-edges", it also consists of edges that specify the affect of congestion. In particular, for each pair of resource nodes α_u and α_v, there is an edge from α_v to α_u if the destination of α_u is the source of α_v. Likewise, the utility functions can be extended appropriately.

Example 5 (Simultaneous Auctions with Budgets)**.** *Previous work has used AGGs to model auctions, including position auctions under generalized first-price and generalized second-price rules (Thompson and Leyton-Brown 2009). Using RGGs, we can use their representations for auctions and extend them to model simultaneous auctions. Moreover, we can include a constraint in the polytope of each player, that ensures the sum of his bids of the simultaneous auctions is no more than his budget. Thus, a player cannot submit bids more than he is able to pay in total.*

Compactness of RGGs RGGs are designed to compactly model scenarios where each player has an exponential number of pure strategies. Other game representations would require exponential spaces. For instance, AGGs have one action node for each pure strategy, using AGGs to model such scenarios would require us to introduce exponentially many action nodes. In (network and local network) congestion games (Example 1, 2 and 4), using AGGs, we need to introduce an action node for each possible subset of resources a player can use. Using RGGs, we only need a node for each resource. In security games (Example 3), using AGGs, we need to introduce node that corresponds to each possible subset of t target locations, while RGGs only need 3 nodes for each target. For simultaneous auctions (Example 5), the space of pure strategies corresponds to the Cartesian product of bid levels for each auction. AGGs would have number of nodes exponential in the number of auctions, while RGG have one node for each bid level in each auction.

4 Equilibrium Computation

Given an RGG, we would like to compute solution concepts such as Nash equilibrium and coarse correlated equilibrium, by leveraging the results of (Chan et al. 2016) as summarized in Section 2. To apply these results, we need to show that the RGG is multilinear, and the subproblems UtilityGradient and PolytopeSolve can be solved in polynomial time.

4.1 Formulating RGGs as Multilinear Games

It is straightforward to verify that congestion games and security games are multilinear. Is every RGG a mulitilinear game? Unfortunately the answer is no. To see this, take an RGG in which the neighborhood of resource α includes itself and another resource β. Consider u^α. To satisfy mutlilinearity, we would like i's utility to be linear in s_i while keeping other players' strategies fixed. However, recall from (1) that the utility contribution from α is $s_{i\alpha}u^\alpha(c^{(\alpha)})$. Let $c^{(\alpha)}_{-i}$ be the configuration induced by other players. This can be written as $s_{i\alpha}[(1-s_{i\beta})u^\alpha(c^{(\alpha)}_{-i}+(1,0))+s_{i\beta}u^\alpha(c^{(\alpha)}_{-i}+(1,1))]$, which is a polynomial (rather than linear) function of s_i.

Extended Formulation If we use a $|S_i|$-dimensional standard simplex to represent the strategy space, then it is straightforward to verify that the game is multilinear. Indeed, all game representations based on this explicit representation of pure strategies, including normal form and AGGs, are multilinear. So we can achieve multilinearity by converting an RGG to this explicit strategy representation, at a cost of an exponential blowup in the dimension of strategy space.

Is there a strategy representation that achieves the best of both worlds: multilinear while having only a polynomial blowup in dimension? The answer is yes for resource graphs of constant in-degree.

The following theorem shows that it is possible to satisfy multilinearity by only adding a polynomial number of nodes, if the graph has constant in-degree.[3] This result bridges the generality and expressiveness of RGGs with the nice computational properties from multilinearity.

Theorem 1. *Given an RGG Γ with maximum in-degree $\mathcal{I}$, we can construct in polynomial time another RGG Γ' with $poly(n, \mathcal{A}, 2^{\mathcal{I}})$ resources, such that (1) there is a bijection between the two RGGs' pure strategies, and the two RGGs are payoff-equivalent; (2) Γ' is multilinear.*

Proof Sketch. We first introduce some notation. Let $s_i^{(\alpha)}$ be the projection of s_i to $\mathbb{Z}_+^{\nu(\alpha)}$, i.e., $s_i^{(\alpha)} = (s_{i\alpha'})_{\alpha'\in\nu(\alpha)}$. Then $s_i^{(\alpha)} \in S_i^{(\alpha)}$, the projection of S_i to $\mathbb{Z}_+^{\nu(\alpha)}$.

Given Γ, for each $i, k \in N$, we will create new resource nodes to make $u_i(s_k, s_{-k})$ linear in s_k (when fixing s_{-k}).

First, we formulate $u_i(s_k, s_{-k})$ as a polynomial function of s_k. Recall that $u_i(s) = \sum_{\alpha\in\mathcal{A}} s_{i\alpha}u^\alpha(c^{(\alpha)})$, so we focus on each term $s_{i\alpha}u^\alpha(c^{(\alpha)})$. If $k = i$, then we have the following polynomial of s_i:

$$s_{i\alpha}u^\alpha(c^{(\alpha)}) = s_{i\alpha}u^\alpha(c^{(\alpha)}_{-i} + s_i^{(\alpha)}) = s_{i\alpha}\sum_{z\in S_i^{(\alpha)}} u^\alpha(c^{(\alpha)}_{-i}+z)\prod_{\alpha'\in\nu(\alpha)} s_{i\alpha'}^{z_{\alpha'}}(1-s_{i\alpha'})^{1-z_{\alpha'}}. \quad (2)$$

After expanding the polynomial function into sum of monomials, each monomial term is of the form $\prod_{\alpha'\in B} s_{i\alpha'}$, for some $B \subset \nu(\alpha)\cup\{\alpha\}$. Since there are at most $2^{|\nu(\alpha)|+1}$ possible such subsets, the number of monomial terms is at most $2^{|\nu(\alpha)|+1}$, and thus u_i can be expressed as a sum of $\sum_\alpha 2^{|\nu(\alpha)|+1}$ monomials of s_i. For $k \neq i$, analogously we can write u_i as a sum of $\sum_\alpha 2^{|\nu(\alpha)|}$ monomials of s_k.

We now create a new resource node for each of these monomials. Specifically, the resources of Γ' consists of the resources of Γ, plus the following additional resource nodes: for each player $k \in N$, each resource $\alpha \in \mathcal{A}$, and each $B \subset \nu(\alpha)\cup\{\alpha\}$, create a resource node β_B^k. Then we define the bijection mapping ϕ from the strategy set S_i of Γ to the new strategy set S_i' of Γ', such that if $s_i' = \phi(s_i)$, then $s'_{i\alpha} = s_{i\alpha}$ for all original resources α, and $s'_{i\beta_B^k} = 0$ if $k \neq i$, and $s'_{i\beta_B^i} = \prod_{\alpha'\in B} s_{i\alpha'}$.

In Γ', utility functions $u'^\alpha = 0$ for the original resources and $u'^{\beta_B^i}$ is equal to the coefficient of monomial $\prod_{\alpha'\in B} s_{i\alpha'}$ in u_i's polynomial expression w.r.t. s_i. This $u'^{\beta_B^i}$ is a function of the monomial terms of other players, and we include the corresponding monomial resource nodes in the neighborhood of β_B^i.

By construction, Γ and Γ' are payoff equivalent, and the utilities of Γ' are multilinear functions of players' pure strategies. To satisfy Condition 2 of multilinearity, we define the new P_i' to be the convex hull of the new S_i'. □

We call Γ' the *multilinear extension* of Γ. We note that Γ' has a larger in-degree than Γ, so a direct representation of the utility functions of Γ' is problematic as their sizes scale exponentially in the in-degree. Nevertheless for our purposes we will not need to explicitly construct the utility functions of Γ'. Instead, as we will show, computational tasks on Γ' can be done implicitly using the resource graph and utility functions of Γ.

Computing UtilityGradient Now that we have mulitilinearity, the next question is whether utility gradients can be computed efficiently. Recall that multilinearity by itself does not imply efficient computation of utility gradients. Direct computation using the RGG's multilinear extension can be inefficient due to its large in-degree. We propose an algorithm that operates on the original RGG's resource graph, which allows us to better exploit the game's utility structure.

Proposition 3. *Given an RGG Γ and its multilinear extension Γ', and a marginal strategy profile π of Γ', the utility gradient $\nabla_k u_i(\pi_{-k})$ can be computed in polynomial time in the size of the utility functions of Γ.*

Proof Sketch. Each π_i of the multilinear extension Γ' includes components for each monomial resource β_B^i, which is the marginal probability that the resources in B are all chosen by i. Using these marginal probabilities we can compute $\Pr(s_i^{(\alpha)}|\pi_i)$ for each $s_i^{(\alpha)} \in S_i^{(\alpha)}$ of Γ, where $s_i^{(\alpha)}$ are the projected strategies as defined in the proof of Theorem 1. The cardinality of $S_i^{(\alpha)}$ is at most $2^{|\nu(\alpha)|}$ so this is still polynomial in the size of utility functions of Γ, and in particular we can efficiently enumerate these projected strategies. Then $u_i(\pi) = \sum_\alpha \sum_{s^{(\alpha)}\in S^{(\alpha)}} s_{i\alpha}u^\alpha(\sum_j s_j^{(\alpha)})\prod_j \Pr(s_j^{(\alpha)}|\pi_j)$,

[3] Since the size of the RGG representation grows exponentially in the in-degree of the graph, RGGs without constant in-degree tend to require exponential. The constant-indegree case captures the interesting class of games from a computational point of view.

which is linear in $\Pr(s_k^{(\alpha)}|\pi_k)$. Since $\Pr(s_k^{(\alpha)}|\pi_k)$ is a linear function of π_k, to compute $\nabla_k u_i(\pi_{-k})$ we just need to compute the coefficients of $\Pr(s_k^{(\alpha)}|\pi_k)$ in the above expression. The problem reduces to an expected utility computation for AGGs, which can be solved in polynomial time by the algorithm in (Jiang, Leyton-Brown, and Bhat 2011). □

PolytopeSolve for the Multilinear Extension Recall from Section 2 that PolytopeSolve is the problem of optimizing an arbitrary linear objective in P_i. When the linear constraints defining P_i are given explicitly, this becomes a linear programming problem and can thus be solved in polynomial time. However, what we need now is PolytopeSolve for the multilinear extension. Since the new S_i' has more dimensions than S_i, the extended polytope $P_i' = conv(S_i')$ is now more complex than the original P_i. In particular, the original constraints of P_i are not sufficient to exactly describe P_i'. Indeed, the convex hull of S_i' may have exponential number of faces, requiring exponential space to specify.

The results of (Chan et al. 2016) allow P_i' to be implicitly represented, as long as we have a polynomial-time algorithm for PolytopeSove. It turns out these P_i' are instances of *correlation polytopes* (Pitowsky 1991), and the linear optimization problem on correlation polytopes is known to be NP-hard even when S_i is all vertices of the hypercube $\{0,1\}^{|\mathcal{A}|}$, or in the case of the simple uniform matroid constraint $\sum_\alpha s_{i\alpha} = K$ (Pitowsky 1991; Xu et al. 2015).

Nevertheless, we can identify subclasses of RGGs for which PolytopeSolve can be solved efficiently. First we observe that PolytopeSolve, a linear optimization problem over P_i' (or S_i'), is equivalent to optimizing a polynomial function over the original S_i. This is by transforming each $s'_{i\beta_B^i}$ in the linear objective to its corresponding monomial $\prod_{\alpha' \in B} s_{i\alpha'}$.

Proposition 4. *Given an RGG Γ with resource graph G, let P_i' be i's polytope in the multilinear extension. Then PolytopeSolve (i.e., linear optimization on P_i') is equivalent to the following POLYOPT(S_i, G) problem: compute $\arg\max_{s_i \in S_i} \sum_\alpha \sum_{B \subset \nu(\alpha)} d_B^\alpha \prod_{\alpha' \in B} s_{i\alpha'}$ given a set of coefficients $\{d_B^\alpha\}_{\alpha,B}$.*

POLYOPT(S_i, G) is equivalent to the problem of computing player i's best response against a pure strategy profile in an RGG with the same players and resource graphs but different utilities from Γ.

Proposition 5. *When the RGG Γ has a resource graph G with bounded treewidth, and the strategy constraints for i are uniform matroid constraints $P_i = \{x \in [0,1]^{|\mathcal{A}|} : \sum_\alpha x_{\alpha \in \mathcal{A}} = K\}$, then there is a polynomial time algorithm for POLYOPT(S_i, G).*

Proof Sketch. The problem is closely related to the following: given a symmetric K-player AGG with action graph G, compute a pure strategy profile with maximum social welfare. The latter problem can be solved in polynomial time for AGGs on bounded treewidth graphs using a dynamic programming approach, see e.g., (Jiang and Leyton-Brown 2007). The only difference is that in our POLYOPT(S_i, G) problem, we additionally require $s_{i\alpha} \leq 1 \ \forall \alpha$. These are unary constraints and can easily be incorporated into the dynamic programming algorithm. □

We can extend the result to a more general class of RGGs using a similar dynamic programming approach.

Proposition 6. *Given an RGG with resource graph G, and player i with polytope $P_i = \{x | D_i x \leq f_i\}$, construct the augmented graph G^{P_i} as follows: add one node j for each constraint $d_{ij}^T x \leq f_i$ of P_i, where d_{ij} is the j-th row of D_i. Then add edge from node j to each resource node α that constraint j refers to, i.e., each α where $D_{ij\alpha} \neq 0$. If G^{P_i} has bounded treewidth and bounded in-degree, POLYOPT(S_i, G) can be computed in polynomial time.*

Taking everything together, we have the following.

Corollary 1. *Given an RGG, if for each player i, the augmented graph G^{P_i} has bounded treewidth and bounded in-degree, then an exact CCE can be computed in polynomial time, and computing a Nash equilibrium is in PPAD.*

4.2 Multilinear RGGs

In Section 4.1, we show how to convert an arbitrary RGG to an (payoff-equivalent) RGG that satisfies multilinearity, incurring a polynomial increase in dimension. We also saw there are subclasses of RGGs, such as congestion games and security games, that are already multilinear without needing to perform any conversion. A natural question is: can we characterize the class of multinlinear RGGs? We provide a sufficient condition for RGGs to be multilinear below. Before stating our proposition, recall that Condition 2 of Definition 1 states that the extreme points of P_i of each player i are integer vectors.

Proposition 7. *Given an RGG $\Gamma = (N, \mathcal{A}, \{S_i\}, G, \{u^\alpha\})$ that satisfies Condition 2 of Definition 1, it is multilinear if for each player i, for each $\alpha \in \mathcal{A}$, and for each $s_i \in S_i$, $\sum_{\alpha' \in \nu(\alpha) \cup \{\alpha\}} s_{i\alpha'} \leq 1$.*

The above proposition follows from Equation 2. The condition in the proposition states that for each player i, it is never the case that two (or more) of her chosen resources are in the same neighborhood. Intuitively, the only obstacle to multilinearity is when two of i's chosen resources are in the neighborhood of α, in which case they potentially influence the utility contribution of α nonlinearly.

Applying this proposition to the examples we saw earlier, the RGG representations of congestion games, security games and simultaneous budgeted auctions (in Examples 2, 3, and 5 respectively) all satisfy this condition and are therefore multilinear. We observe that in these games, a player's sub-decisions can still influence each other via the strategy constraints, and there can be a wide variety of utility influence across different players; it is the lack of utility influence *within* a player's sub-decisions that ensures multilinearity.

One issue is whether it is easy to check if this condition holds for a given Γ. Directly checking for each s_i would involve enumerating the set of pure strategies. Instead, we can verify whether the above condition holds for each player using linear programming. In particular, for each $i \in N$ and $\alpha \in \mathcal{A}$, consider the program: $\max \sum_{\alpha' \in \nu(\alpha) \cup \{\alpha\}} s_{i\alpha'}$

such that $D_i s_i \leq f_i$ and $s_i \in \{0,1\}^{|A|}$. Since $P_i = \{x \in \mathbb{R}^m | D_i x \leq f_i\} = Conv(S_i)$, we can relax the constraint on s_i and replace it with $0 \leq s_{i\alpha} \leq 1$ for all $\alpha \in \mathcal{A}$. As a result, we have a linear program, and its solution can be used to verify whether an RGG is multilinear.

5 Conclusion and Open Problems

In this paper, we introduce *Resource Graph Games* (RGGs), a general compact representation for games with structured strategy spaces. Not only can RGGs capture all of the games studied in the literature, they can also be used to model new game-theoretic problems that cannot be modeled by other compact representations. We provide efficient computation of solution concepts given RGGs by establishing a connection between RGGs and multilinear games. In particular we show that for some classes of RGGs, expected utilities, best responses, and coarse correlated equilibrium can be computed efficiently, and computation of Nash equilibrium is in PPAD. An open problem is practical algorithms for Nash equilibrium. Given the computability of the best responses, a natural direction is to adapt best-response-based heuristics such as *fictitious play* dynamic for computing Nash equilibrium in RGGs.

References

Ahmadinejad, A.; Dehghani, S.; Hajiaghayi, M.; Lucier, B.; Mahini, H.; and Seddighin, S. 2016. From Duels to Battlefields: Computing Equilibria of Blotto and Other Games. In *AAAI: Proceedings of the AAAI Conference on Artificial Intelligence*.

Chan, H.; Jiang, A. X.; Leyton-Brown, K.; and Mehta, R. 2016. Multilinear games. In *WINE: Proceedings of The 12th Conference on Web and Internet Economics*.

Daskalakis, C.; Fabrikant, A.; and Papadimitriou, C. 2006. The game world is flat: The complexity of Nash equilibria in succinct games. In *ICALP: Proceedings of the International Colloquium on Automata, Languages and Programming*, 513–524.

Dunne, P. E., and Wooldridge, M. 2012. Towards tractable boolean games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 939–946. International Foundation for Autonomous Agents and Multiagent Systems.

Harrenstein, P.; van der Hoek, W.; Meyer, J.-J.; and Witteveen, C. 2001. Boolean games. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*, 287–298. Morgan Kaufmann Publishers Inc.

Immorlica, N.; Kalai, A. T.; Lucier, B.; Moitra, A.; Postlewaite, A.; and Tennenholtz, M. 2011. Dueling algorithms. In *STOC: Proceedings of the Annual ACM Symposium on Theory of Computing*.

Jiang, A. X., and Leyton-Brown, K. 2007. Computing pure Nash equilibria in symmetric Action-Graph Games. In *AAAI: Proceedings of the AAAI Conference on Artificial Intelligence*, 79–85.

Jiang, A. X.; Leyton-Brown, K.; and Bhat, N. 2011. Action-graph games. *Games and Economic Behavior* 71(1):141–173.

Kakade, S.; Kearns, M.; Langford, J.; and Ortiz, L. 2003. Correlated equilibria in graphical games. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, 42–47. New York, NY, USA: ACM.

Kearns, M.; Littman, M.; and Singh, S. 2001. Graphical models for game theory. In *UAI: Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 253–260.

Koller, D., and Milch, B. 2001. Multi-agent influence diagrams for representing and solving games. In *IJCAI: Proceedings of the International Joint Conference on Artificial Intelligence*.

Köppe, M.; Ryan, C. T.; and Queyranne, M. 2011. Rational generating functions and integer programming games. *Oper. Res.* 59(6):1445–1460.

Korzhyk, D.; Conitzer, V.; and Parr, R. 2010. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*.

Nguyen, T.-V.-A., and Lallouet, A. 2014. A complete solver for constraint games. In O'Sullivan, B., ed., *Principles and Practice of Constraint Programming*, volume 8656 of *Lecture Notes in Computer Science*. Springer International Publishing. 58–74.

Nguyen, T.-V.-A.; Lallouet, A.; and Bordeaux, L. 2013. Constraint games: Framework and local search solver. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, 963–970.

Ortiz, L., and Kearns, M. 2003. Nash propagation for loopy graphical games. In *NIPS: Proceedings of the Neural Information Processing Systems Conference*, 817–824.

Papadimitriou, C., and Roughgarden, T. 2008. Computing correlated equilibria in multi-player games. *Journal of the ACM* 55(3):14.

Pitowsky, I. 1991. Correlation polytopes: Their geometry and complexity. *Math. Program.* 50(3):395–414.

Rabinovich, Z.; Gerding, E.; Polukarov, M.; and Jennings, N. R. 2009. Generalised fictitious play for a continuum of anonymous players. In *IJCAI: Proceedings of the International Joint Conference on Artificial Intelligence*, 245–250.

Tambe, M. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

Thompson, D. R. M., and Leyton-Brown, K. 2009. Computational analysis of perfect-information position auctions. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, EC '09, 51–60.

Vorobeychik, Y. 2008. *Mechanism Design and Analysis Using Simulation-Based Game Models*. Ph.D. Dissertation, University of Michigan.

Xu, H.; Jiang, A. X.; Sinha, A.; Rabinovich, Z.; Dughmi, S.; and Tambe, M. 2015. Security games with information leakage: Modeling and computation. In *AAAI*.