An FPTAS for Computing Nash Equilibrium in Resource Graph Games, with Applications to Security Games, Congestion Games, and Bilinear Games

Hau Chan
Department of Computer Science and Engineering
University of Nebraska-Lincoln, Lincoln, NE
hchan3@unl.edu

Albert Xin Jiang
Department of Computer Science
Trinity University, San Antonio, TX
xjiang@trinty.edu

Abstract

We consider the computational complexity of computing a mixed-strategy Nash equilibrium (MSNE) in resource graph games (RGGs) [Jiang, Chan, Leyton-Brown 2017], a compact representation for games with an exponential number of strategies. In a resource graph game, there are m resources and a directed graph of the resources, and each player's pure strategy set consists of subsets of resources. Each player's pure strategy is represented by a binary vector, and the pure strategy set is represented compactly using a rational polytope defined by a set of linear inequality constraints. Given the pure strategies of the players, each player's utility depends on the directed resource graph and the numbers of times the neighboring resources are used.

In this paper, we provide the first FPTAS for computing an MSNE in any symmetric multilinear RGG where its constraint moralized resource graph (a graph formed between the moralized resource graph and the constraints defining the strategy polytope) has bounded treewidth. Our FPTAS can be generalized to a constant number of player types. As a consequence, our FPTAS provides the first and improved approximation results for domain specific games such as single-attacker and single-defender security games and congestion games. Finally, leveraging the RGG representation and our FPTAS, we obtain an FPTAS to compute an MSNE for a large class of bilinear games.

1 Introduction

There has been increasing interest in using game theory to model real-world 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. 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 [18], multi-agent influence diagrams [19] and action-graph games [16]; as well as work on efficient algorithms for computing solution concepts such as Nash equilibrium [25, 7] and (coarse) correlated equilibrium [17, 26] 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 workers to tasks, 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. Several classes of multi-player game models studied in the recent literature have such *structured strategy spaces*, including network congestion games [7], simultaneous auctions and other multi-item auctions [30, 27], dueling algorithms [13], integer programming games [20], Blotto games [1], and security games [21, 29]. These authors proposed compact game representations that are suitable for their specific domains and computation needs.

More recently, [14] proposed resource graph games (RGGs), a general compact representation for games with structured strategy spaces that unifies and generalizes many of the existing classes of structured games mentioned above. 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. In more detail, in an RGG, there are n players and a set $\mathcal{A}$ of resources. A pure strategy of a player is a subset of the resources, represented by an $|\mathcal{A}|$-dimensional 0-1 vector where 1 in the j^{th} dimension denotes the usage of the j^{th} resource. Each player's set of pure strategies is represented compactly as a *polytopal strategy space*: the set of integer points in a convex polytope, which are specified by a set of linear inequality constraints. There is a *resource graph*, a directed graph whose vertices are the resources $\mathcal{A}$, and the graph depicts the utility structure of the game. There is a utility contribution for using each resource, specified by a *local utility function*, which depends on the number of times each of its neighboring resources are used by the players. Given the pure strategies of the players, a player's utility is the sum of the utility contributions of the resources that the player uses in his/her pure strategy.

In this paper we consider the problem of computing a mixed-strategy Nash equilbrium (MSNE) in RGGs. RGGs can represent arbitrary games, as a result the PPAD-hardness of finding an MSNE in normal form games [6, 8] implies the PPAD-hardness for finding MSNE in RGGs. In the other direction, [14] showed that for certain subclasses of RGGs, the problem 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.

1.1 Our Contributions

A natural question arises: are there subclasses of RGGs for which MSNE can be computed (or approximated) in polynomial time? In this paper we provide one answer to this question by identifying a subclass of RGGs that admits a fully polynomial time approximation scheme (FPTAs) for finding MSNE. In particular, we focus on symmetric RGGs that satisfy a multilinearity condition (the utility functions are linear in each player's strategy vector while keeping others' strategies fixed). We define constraint moralized resource graph as an undirected graph constructed by starting from the moralized graph (also known as the primal graph) of the resource graph, then adding nodes corresponding to the linear strategy constraints, and edges connecting each constraint with the resources that are involved in the constraint. Our main result is summarized as follows.

Main Theorem (Informal). There is a FPTAS for computing MSNE for a symmetric multilinear RGG whose constraint moralized resource graph has bounded treewidth.

Our FPTAS can be extended to find specific MSNE, such as MSNE with the greatest (or lowest) social welfare. We also generalize our FPTAS to a constant number of player types.

We then applied our FPTAS to classes of RGGs that correspond to games studied in literature, and obtained new and improved results for these games. In particular, we show FPTAS for computing MSNE in

- generalizations to security games. Initial studies on security games focused on the case of multiple defender units and the attacker choosing a single target. Efficient algorithms for finding Stackelberg equilibria were proposed, and [23] showed that under mild conditions, these Stackelberg strategies are often also Nash equilibrium strategies. [22] studied security games in which the attacker can choose multiple targets, and presented a polynomial-time algorithm for finding a Nash equilibrium when defender's and attacker's strategy sets are uniform matroids (i.e., a single capacity constraint that bounds the total number of resources chosen). We show that for both the single attack model of [23] and the multiple attack model of [22] the moralized constraint resource graphs of corresponding RGG representations have constant treewidth, and therefore our FPTAS can be applied. Our FPTAS can be applied to more general cases of players' strategy constraints not covered by existing results, as long as the constraint matrices are totally unimodular and the treewidth of the resulting moralized constraint resource graph is bounded.

- congestion games. Congestion games model the situation in which there is a set of resources, and each player selects a subset of resources. The cost of using a resource depends on the total number of players using it. As such, the player's goal is to select a subset of resources that minimizes the total cost. A special type of congestion games is those where the players' strategy constraints are totally unimodular. Our FPTAS provides a more general (bounded treewidth) graphical structure to cover a boarder range of this type of congestion games than the FPTAS in [4].

- bilinear games [12], including bimatrix games. These are two-player games whose utilities are characterized by payoff matrices A and B. When A and B are sparse matrices, the RGG representations have nontrivial graph structure. We can apply our FPTAS when the constraint moralized resource graph has bounded treewidth. Furthermore, due to the special structure of bilinear game utilities, we can avoid the moralization step, resulting in further reduced treewidth.

1.2 Main Technical Ideas

One immediate obstacle is that representing mixed strategies explicitly would require exponential space. The multi-linearity of the game allows us to represent mixed strategies compactly using marginal vectors (marginal probabilities that each resource is being chosen), with dimension linear in the number of resources. We thus search for approximate MSNE in the space of marginal vectors.

The standard formulation for Nash equilibrium consists of inequality constraints for each pure strategy. For games with exponential numbers of pure strategies, such a formulation is inconvenient as it requires reasoning about an exponential number of constraints. Instead, we start with a more efficient formulation for MSNE, based on the primal and dual linear programs for computing best response in RGGs. The resulting formulation for MSNE has a polynomial number of variables (the marginal vector and the dual variables), and a polynomial number of constraints (the primal feasibility, dual feasibility, and strong duality constraints). Our task is thus to solve this polynomial-sized (nonlinear) feasibility problem.

We solve the feasibility problem by exploiting the independence structure among its variables and constraints. In particular, expected utility computation (needed by e.g., the strong duality condition) only depends on neighboring nodes, and each strategy constraint might only involve certain subset of resources. This structure is encoded as the constraint moralized resource graph.

In a standard graph-based approach for constraint satisfaction, a moralization step is usually required, in which the variables associated with each k-ary constraint is connected together into a clique before applying tree decomposition. In our algorithm, we are doing the moralization for edges corresponding to the resource graph (representing utility dependence), in order to make sure the resulting tree decomposition has enough information to carry out the expected utility computation. On the other hand, we do not need to do moralization for the other edges between resources and constraints. Although they correspond to k-ary constraints, those constraints are linear, as a result, instead of moralizing and then process the k-ary constraint all at once, we could keep track of partial sums of the constraint. We keep partial sums for various components of our primal feasibility, dual feasibility and strong duality constraints. By avoiding moralizations we potentially reduce the treewidth of the resulting graph.

After discretizing the variables and the partial sums, we propose a message passing algorithm that operates on a tree decomposition of the constraint moralized resource graph. Each resource node in the graph is associated with its marginal variable and partial sum for the corresponding row in dual constraints; each constraint node in the graph is associated with partial sum for that row of the strategy constraint and its corresponding dual variable. The algorithm first pass messages from leaves up, ending at root, with messages containing the feasible values for variables and partial sums that are relevant to the rest of the graph. It then pass messages from root down, ending at leaves, to construct an ϵ-MSNE.

If the local utility functions are linear, i.e., the sum of contributions from each neighbor, then we can again apply the above partial sum idea and avoid needing to moralize the resource graph edges as well, further reducing the treewidth. This underlies our results for bilinear games, whose RGG representations have linear local utilities.

1.3 Related Work

As mentioned earlier, RGGs are introduced by [14] to compactly model games with an exponential number of actions. For some RGGs that satisfy multilinearity, it has been shown that computing an exact coarse correlated equilibrium can be computed in polynomial time and computing a mixed-strategy Nash equilibrium (MSNE) is in PPAD using the results of [5].

A special case of RGGs is congestion games [28] where the nodes in the resource graphs have only self-edges. It is known that every congestion game has a pure-strategy Nash equilibrium (PSNE) and even finding a PSNE in network congestion games is PLS-complete [11]. However, there is a polynomial time to find a PSNE in symmetric network congestion games. More recent works [4, 10] study congestion game settings where the strategy spaces of the players can be compactly represented using polytopes defined by sets of linear inequalities. When the polytopes have totally unimodular properties, there is a polynomial time algorithm to compute a PSNE in congestion games [10] and there is a fully polynomial time approximation scheme (FPTAS) to compute a mixed-strategy Nash equilibrium (MSNE) [4].

Action graph games (AGGs) [16] are also special cases of RGGs in which the resources are the explicit actions of the players. While determining whether there is a PSNE in AGGs is NP-hard [15, 9] and computing approximate MSNE in AGGs is PPAD-hard [9], for AGGs with bounded degree and treewidth action graphs and constant number of player types, there is an FPTAS for computing a MSNE [9].

A bilinear game [12] is a two-player game whose utilities are characterized by payoff matrices A and B. [12] show that there are polynomial algorithms to compute MSNE when the sum of the rank of $A + B$ is one and the rank of A or B is constant and an FPTAS for computing MSNE when the rank of $A + B$ is constant.

2 Preliminaries

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

2.1 Games, Strategies and Equilibrium

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 $u_i : S \to \mathbb{R}$.

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.

For $\epsilon \geq 0$, player i's mixed strategy σ_i is an ϵ-best response to σ_{-i} if $u_i(\sigma_i, \sigma_{-i}) \geq \max_{\sigma'_i\in\Sigma_i} u_i(\sigma'_i, \sigma_{-i}) - \epsilon$. An equivalent condition is $u_i(\sigma_i, \sigma_{-i}) \geq u_i(s'_i, \sigma_{-i}) - \epsilon \ \forall s'_i \in S_i$. A mixed strategy profile σ is an ϵ-MSNE if for each player $i \in N$, σ_i is an ϵ-best response to σ_{-i}. A best response is a 0-best response and an MSNE is a 0-MSNE.

2.2 Resource Graph Games

A resource graph game (RGG) is specified by the tuple $\Gamma = (N, \mathcal{A}, \{S_i\}_{i=1,\ldots,n}, G, \{u^\alpha\}_{\alpha\in\mathcal{A}})$, where:

- $N = \{1, 2, ..., n\}$ is the set of n players.
- $\mathcal{A} = \{1, ..., m\}$ is the set of m resources.
- S_i is a nonempty set of pure strategies for player i, and each $s_i \in S_i$ is represented by an $|\mathcal{A}|$-dimensional 0-1 vector.
- S_i is represented as a *polytopal strategy space* where $S_i = P_i \cap \{0, 1\}^{|\mathcal{A}|}$, $P_i = \{x \in [0, 1]^{|\mathcal{A}|} | D_i x \leq f_i\}$, $D_i \in \mathbb{Z}^{l_i\times|\mathcal{A}|}$, and $f_i \in \mathbb{Z}^{l_i}$. In other words, P_i is a rational polytope defined by l_i linear constraints. We let $s_{i\alpha}$ to denote the component corresponds to $\alpha \in \mathcal{A}$.
- Given a pure-strategy profile $s = (s_1, ..., s_n)$, the configuration $c = \sum_{i\in N} s_i$ is an integer vector that denotes the total number of players who have selected each resource.
- The resource graph $G = (\mathcal{A}, E)$ is a directed graph that could contain self-loops. The neighborhood of $\alpha \in A$ is denoted by $\nu(\alpha)$ (which could include α). We let $C^{(\alpha)} \subset \mathbb{Z}_+^{\nu(\alpha)}$ to be the set of possible local configurations over $\nu(\alpha)$, and $c^{(\alpha)} \in C^{(\alpha)}$ to be a local configuration.
- The local utility function $u^\alpha : C^{(\alpha)} \to \mathbb{R}$ represents the utility contribution of using α given the configuration over its neighborhood $\nu(\alpha)$.
- Given the pure-strategy profile $s = (s_1, ..., s_n)$, the utility of player i is defined to be

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

[14] showed that the size of this representation is polynomial in m, $n^{\mathcal{I}}$, and $\sum_i l_i$, where $\mathcal{I}$ is the maximum in-degree of the resource graph.

2.3 Multilinearity and Marginal Represntation of Mixed Strategies

When $|S_i|$ is exponential, representing a mixed strategy σ_i explicitly would take exponential space. Given player i's mixed strategy σ_i, the *marginal vector* π_i that corresponds to σ_i is $\pi_i = E_{\sigma_i}[s_i] = \sum_{s_i \in S_i} \sigma_i(s_i) s_i$. In other words, $\pi_{i\alpha}$ is the marginal probability that resource α is chosen under the mixed-strategy σ_i. Thus if we represent a mixed strategy using its marginal vector, this would only require $O(m)$ space. However, for this representation to work, we need to be able to express the expected utilities of the game in terms of marginal vectors. [5] showed that this can be done if the game has the *multilinear* property: for all $i, j \in N$, given a fixed s_{-j}, u_i is a linear function of s_j. Under this property, each marginal vector corresponds to an infinite number of mixed strategies, all payoff-equivalent to each other.

Not all RGGs are multilinear; the following proposition gives a sufficient condition for an RGG to be multilinear.

Proposition 1 (Proposition 7 of [14])**.** *An RGG, $\Gamma = (N, \mathcal{A}, \{S_i\}_{i=1,\ldots,n}, G, \{u^\alpha\}_{\alpha\in\mathcal{A}})$, is multilinear if, for each player i, for each $\alpha \in \mathcal{A}$, and for each $s_i \in S_i$, $\sum_{\alpha' \in \nu(\alpha)\cup\{\alpha\}} s_{i\alpha'} \leq 1$. Moreover, given an RGG, we can verify the above condition in polynomial time.*

For other RGGs, [14] showed that they can be transformed to a payoff-equivalent mutilinear game with polynomial increase in the dimensionality of the strategy spaces. Nevertheless, for this paper we will focus on RGGs satisfying the condition for Proposition 1.

In practice, it is often desirable to be able to recover a mixed strategy from a marginal vector representation. Since the explicit representation of mixed strategies has exponential size, we would instead like to have the mixed strategy specified in some implicit manner that requires less space, e.g., as a sparse vector or an efficient sampling scheme. [5] provides one sufficient condition for the efficient recovery of mixed strategies from marginals: If the constraint matrix D_i is totally unimodular, then given a marginal vector π_i, we can efficiently generate a mixed strategy σ_i, represented as a sparse vector with polynomial number of nonzero entries, such that $\pi_i = \sum_{s_i \in S_i} \sigma_i(s_i) s_i$.

In summary, to ensure the compact representation of mixed strategies as marginal vectors, we make the following assumptions:

Assumption 1. *For all i, the constraint matrix D_i is totally unimodular.*

Assumption 2. *For each $i \in N$, for each $\alpha \in \mathcal{A}$, and for each $s_i \in S_i$, $\sum_{\alpha' \in \nu(\alpha)\cup\{\alpha\}} s_{i\alpha'} \leq 1$.*

Given Assumption 1, we claim that without loss of generality we can assume $f_i \in [-m, m]^{l_i}$. To see this, we note that entries of a totally unimodular D_i must be either 0, 1, or -1. Since strategies are 0-1 vectors, $D_i x \in [-m, m]^{l_i}$. Then if $f_{ij} < -m$, the corresponding row of the constraints is unsatisfiable, resulting in an empty polytope P_i, which violates our assumption that S_i is nonempty. If $f_{ij} > m$, the corresponding row of constraints is always satisfied, and we could change f_{ij} to m without changing the polytope P_i.

Assumption 3. *For all $\alpha \in \mathcal{A}$, the range of u^α is bounded in* $[0, 1]$.

A fully-polynomial time approximation scheme (FPTAS) for MSNE is an algorithm that given an RGG, computes an ϵ-MSNE in time poly$(n^{\mathcal{I}}, m, l, \frac{1}{\epsilon})$. While Nash equilibria are preserved under affine transformations of the utility functions u_i, ϵ-Nash equilibria are not. E.g. multiplying all u_i by a constant κ would turn a ϵ-MSNE into a $\kappa\epsilon$-MSNE in the new game. A standard practice is to normalize the utility functions u_i into the range $[0, 1]$, by the affine mapping $u_i \mapsto (u_i - u_{min})/(u_{max} - u_{min})$, where $u_{max} = \max_{i,s} u_i(s)$ and $u_{min} = \min_{i,s} u_i(s)$ are the maximum and minimum achievable utilities of the game, respectively. Then use the normalized game to measure the quality of approximation, i.e., ϵ. An ϵ-MSNE in the original game would correspond to an $\frac{\epsilon}{(u_{max} - u_{min})}$-MSNE in the normalized game. While Assumption 3 implies $u_i(s) \in [0, m]$ for all i and s, these are in general not tight bounds on u_{max} and u_{min}, and indeed u_{max} and u_{min} may be hard to compute given an RGG. We note that this is a common issue with compact game representations whose utility functions are sums of other functions, e.g., polymatrix games [3]. For our purposes, in order to ensure that our FPTAS remains an FPTAS for the normalized game, it is sufficient to assume that $(u_{max} - u_{min})$ is lower-bounded by an inverse polynomial of the size of the game.

Assumption 4. $(u_{max} - u_{min}) \geq \frac{1}{poly(n^{\mathcal{I}}, m, \sum_i l_i)}$.

In practice this is a reasonable assumption for natural classes of games, as we would normally expect $u_{max} - u_{min}$ to not decrease as the size of the game grows. Indeed it can be easily verified that this holds for all example games discussed in this paper.

3 Computing Approximate MSNE

We begin by re-expressing and rewriting Equation 1 in terms of marginal vectors. The reformulation allows us to derive some important properties of Nash equilibrium.

3.1 A Useful Expression of Expected Utilities

Before stating our results, we define some notations. For each $\alpha \in \mathcal{A}$, let $\Omega_{j\alpha} = \{\omega \in \{0,1\}^{|\nu(\alpha)|} \mid \sum_{i=1}^{|\nu(\alpha)|} \omega_i \leq 1\}$ be the set of local configurations that specifies player j's resource usage in the neighborhood of $\nu(\alpha)$. We also let $\Omega_{-i,\alpha} = \prod_{j \in N\setminus\{i\}} \Omega_{j\alpha}$. For simplicity, we assume that, for $\alpha \in \mathcal{A}$, $\nu(\alpha)$ is ordered lexicographically, and the k^{th} element in $\nu(\alpha)$ corresponds to ω_k and s_{jk} for each $\omega \in \Omega_{j\alpha}$, for each $s_j \in S_j$, and for each player j.

Proposition 2. *Given a multilinear RGG that satisfies the conditions in Proposition 1, a mixed-strategy profile $\sigma_{-i} = (\sigma_j)_{j\in N\setminus\{i\}}$ and its corresponding marginal vectors $\pi_{-i} = (\pi_j)_{j\in N\setminus\{i\}}$, the expected utility of i playing $s_i \in S_i$ is*

$$u_i(s_i, \pi_{-i}) = \sum_{\alpha\in\mathcal{A}} \sum_{\omega_\alpha \in \Omega_{-i,\alpha}} s_{i\alpha} u^\alpha \left(\sum_{j\in N\setminus\{i\}} \omega_{j\alpha} + s_i^{(\alpha)} \right) \prod_{j\in N\setminus\{i\}} Pr(w_{j\alpha}|\pi_j),$$

where $s_i^{(\alpha)} = (s_{i\alpha'})_{\alpha'\in\nu(\alpha)}$ is the projection of s_i to $\mathbb{Z}_+^{\nu(\alpha)}$.

Proof. Following from Equation (1), we can express the expected utility of player i as the sum over the configurations. We have that

$$\begin{aligned} u_i(s_i, \sigma_{-i}) &= \sum_{\alpha\in\mathcal{A}} \sum_{s_{-i}\in S_{-i}} s_{i\alpha} u^\alpha \left(c^{(\alpha)}\right) \prod_{j\in N\setminus\{i\}} \sigma_j(s_j) \\ &= \sum_{\alpha\in\mathcal{A}} \sum_{s_{-i}\in S_{-i}} s_{i\alpha} u^\alpha \left(\sum_{j\in N} s_j^{(\alpha)}\right) \prod_{j\in N\setminus\{i\}} \sigma_j(s_j) \\ &= \sum_{\alpha\in\mathcal{A}} \sum_{s_{-i}^{(\alpha)}\in S_{-i}^{(\alpha)}} s_{i\alpha} u^\alpha \left(\sum_{j\in N} s_j^{(\alpha)}\right) \prod_{j\in N\setminus\{i\}} Pr\left(s_j^{(\alpha)}\right), \end{aligned}$$

where the second equality is by the definition of configuration and the third equality is because we only need to keep track of the strategies of the neighborhood of α. Our claim follows from the last equality because each player can only use at most one resource in the neighborhood (Proposition 1 (b)). Thus, we can replace $s_j^{(\alpha)}$ and $S_j^{(\alpha)}$ by $\omega_{j\alpha}$ and $\Omega_{j\alpha}$, respectively. Finally, the probability of a player of using a particular resource in the local neighborhood is just its marginal probability. □

Our next lemma follows from the above proposition.

Lemma 1. *Given a multilinear RGG that satisfies the conditions in Proposition 1, a mixed-strategy profile $\sigma = (\sigma_j)_{j\in N}$ and its corresponding marginal vectors $\pi = (\pi_j)_{j\in N}$, the expected utility of i is*

$$u_i(\pi_i, \pi_{-i}) = \sum_{\alpha\in\mathcal{A}} \pi_{i\alpha} \sum_{\omega_\alpha\in\Omega_{-i,\alpha}} u^\alpha \left(\sum_{j\in N\setminus\{i\}} \omega_{j\alpha} + \omega_{i\alpha} \right) \prod_{j\in N\setminus\{i\}} Pr(w_{j\alpha}|\pi_j),$$

where $w_{i\alpha}$ is all zero and has a value of one at the α entry if $\alpha \in \nu(\alpha)$.

Proof. From the above proposition, we have

$$\begin{aligned}
u_i(\sigma_i, \pi_{-i}) &= \sum_{\alpha \in \mathcal{A}} \sum_{s_i \in S_i} \sum_{\omega_\alpha \in \Omega_{-i,\alpha}} s_{i\alpha} u^\alpha \left(\sum_{j \in N \setminus \{i\}} \omega_{j\alpha} + s_i^{(\alpha)} \right) \prod_{j \in N \setminus \{i\}} Pr(w_{j\alpha} | \pi_j) \sigma_i(s_i) \\
&= \sum_{\alpha \in \mathcal{A}} \sum_{s_i^{(\alpha)} \in S_i^{(\alpha)}} \sum_{\omega_\alpha \in \Omega_{-i,\alpha}} s_{i\alpha} u^\alpha \left(\sum_{j \in N \setminus \{i\}} \omega_{j\alpha} + s_i^{(\alpha)} \right) \prod_{j \in N \setminus \{i\}} Pr(w_{j\alpha} | \pi_j) Pr(s_i^{(\alpha)}) \\
&= \sum_{\alpha \in \mathcal{A}} \pi_{i\alpha} \sum_{\omega_\alpha \in \Omega_{-i,\alpha}} u^\alpha \left(\sum_{j \in N \setminus \{i\}} \omega_{j\alpha} + \omega_{i\alpha} \right) \prod_{j \in N \setminus \{i\}} Pr(w_{j\alpha} | \pi_j),
\end{aligned}$$

where the first equality is using the above proposition to compute the expected utility of player i, the second equality is by projecting the strategies to their local neighborhood α, and the third equality is from the fact that the sum is zero if $s_{i\alpha} = 0$ and the sum is nonzero with the marginal probability of player i using α, which is $\pi_{i\alpha}$. Finally, depending on whether $\alpha \in \nu(\alpha)$, we need to increment the count of the α by one in the configuration. $\square$

3.2 An Efficient Primal-Dual Definition of MSNE

From Lemma 1, we can express the expected utility of marginal vectors as $\pi_i^T \nabla_i(\pi_{-i}) = u_i(\pi, \pi_{-i})$ where $\nabla_i(\pi_{-i})$ is the utility gradient of i and $\nabla_i(\pi_{-i})_\alpha$ denotes the expected utility contribution of i from using the resource α and π_i^T is the transpose of π_i. [14] showed that given an RGG and π_{-i}, the utility gradient $\nabla_i(\pi_{-i})$ can be computed in polynomial time. Moreover, to compute $\nabla_i(\pi_{-i})_\alpha$ we only need to know $(\pi_{j\alpha'})_{j \neq i, \alpha' \in \nu(\alpha)}$, i.e., other players' marginals projected to α's neighborhood.

Notice that in the standard definition of MSNE, we need to enumerate all pure strategies to verify whether a mixed-strategy profile is a MSNE. Clearly, this is not feasible for RGGs or games with exponential number of pure strategies. Therefore, we need an alternative and efficient way of checking whether a mixed strategy profile is an MSNE.

Best-responses of the Players In particular, given the marginal vectors of other players π_{-i}, the best-response for player i is the solution to the following linear program.

Primal

$$\begin{aligned}
\text{maximize} \quad & \pi_i^T \nabla_i(\pi_{-i}) \\
\text{subject to} \quad & D_i \pi_i \leq f_i \\
& \pi_{i\alpha} \geq 0 \text{ for } \alpha \in \mathcal{A}
\end{aligned}$$

Here we assume that the constraint $\pi_{i\alpha} \leq 1$ is embedded into the constraints $D_i \pi_i \leq f_i$. We can formulate the dual linear program as the following.

Dual

$$\begin{aligned}
\text{minimize} \quad & f_i^T \lambda_i \\
\text{subject to} \quad & D_i^T \lambda_i \geq \nabla_i(\pi_{-i}) \\
& \lambda_{ij} \geq 0 \text{ for } j = 1, 2, ..., l_i
\end{aligned}$$

Given that the primal LP has a solution, the primal and dual LPs solutions have the same optimal objective value. As a result, π_i is i's best response when there exists a feasible dual vector λ_i such that $\pi_i^T \nabla_i(\pi_{-i}) = f_i^T \lambda_i$. Thus, a feasible marginal vector $\bar{\pi}$ is an MSNE if for each player i, there exists $\lambda'_{ij}s \geq 0$ such that $D_i^T \lambda_i \geq \nabla_i(\bar{\pi}_{-i})$ and $\bar{\pi}_i^T \nabla_i(\bar{\pi}_{-i}) = f_i^T \lambda_i$.

Notice that this primal-dual formulation of best responses do not tell us how to find an MSNE; in particular we cannot simply solve the LPs to find an MSNE because these are LPs only for fixed π_{-i}.

3.3 Symmetric Multilinear RGGs

We consider computing an optimal approximate Nash equilibrium in symmetric multilinear RGGs that satisfy Assumptions (1-4). This is an important class of RGGs, and it contains symmetric action graph games and (network) congestion games. We define symmetric RGGs to be those games in which the players have the same polytopal strategy space.

Definition 1. *An RGG, $\Gamma = (N, \mathcal{A}, \{S_i\}_{i=1}^n, G, \{u^\alpha\}_{\alpha\in\mathcal{A}})$, is symmetric if and only if for all player i, $D_i = D$, $f_i = f$, $S_i = S$, and $P_i = P$ for some $D \in \mathbb{Z}^{l_i \times |\mathcal{A}|}$ and $f \in \mathbb{Z}^{l_i}$.*

It turns out that every symmetric game has a symmetric MSNE in which every player plays the same (possibly mixed) strategies [24]. As a result, we have the following.

Proposition 3. *For every symmetric multilinear RGGs, there is a symmetric MSNE in which every player plays the same strategy.*

As mentioned earlier, we will focus on computing marginal vectors. The above proposition implies that there is a marginal representation of symmetric Nash equilibrium.

Next, we show the existence of a symmetric ϵ-MSNE in which the marginal vector lies in a discretized grid in P_i. This allows us to search for an ϵ-MSNE in this discretized space.

Lemma 2. *For any $\delta > 0$, there is a (symmetric) marginal vector $\forall i\; q_i = q = (q_\alpha)_{\alpha\in\mathcal{A}}$ in which $q_\alpha \in \{0, \frac{1}{K}, \frac{2}{K}, \dots, 1\}$ and $K = O(\frac{\log m}{\delta^2})$, such that $u(q_i, q_{-i}) = q^T\nabla(q_{-i}) \geq q'^T\nabla(q_{-i}) - O(\delta dnm)$ for $q' \in [0,1]^m$ that satisfies $Dq' \leq f$ and d is the maximum in-degree of the resource graph G.*

Proof Sketch. The proof is similar to the proof of Lemma 2 in [4]. The key idea is to compare i's expected utility under the symmetric marginal representation of the Nash equilibrium (from Proposition 5) and i's expected utility under q (its existence is guaranteed by the approximated version of the Caratheodory's theorem (Theorem 3 of [2])) using the formula in Lemma 1. The difference of the two expected utilities of each resource is roughly the difference of two multinomial terms/distributions with at most $d+1$ events. It can be shown that its total variation distance is $O(\delta dn)$ [9]. Since we have m resource, the total variation distance is $O(\delta dnm)$. Using the total variation distance, we can derive our claim. $\square$

The above lemma implies that if every player plays according to q, then we have an $O(\delta dnm)$-MSNE. Our next step is to present an efficient algorithm to find such q.

3.4 An Efficient Algorithm to Compute ϵ-MSNE

From Lemma 2, there is a marginal vector q that lies in the m-dimensional uniform discretized grid with discretization size $\frac{1}{O(\delta dnm)}$. Thus, a brute-force approach does not yield an efficient algorithm. Our approach is based on a similar principle of [4] where we use the definition of NE in terms of best-response and linear programming of Section 3.2 and introduce an algorithm that run on some variant of the resource graph (will be discussed in Section 3.4.3). The idea is to keep track of the possible partial $\{q_\alpha\}_{\alpha\in A'}$ for some set A' that satisfies the linear programs and can be potentially be used to form the ϵ-MSNE. At the same time, we need to keep track of the partial utility sums (i.e., primal solution), partial constraint sums (for each of the constraints in the dual/primal LPs), and partial sums of the dual solution.

However, the possible values of the partial utility sums can grow exponentially. To make our computational task tractable, we discretize the utility space. Such additional discretization will add a small approximation error to the ϵ-MSNE from our algorithm. Below, we provide the discretization and the error bound.

3.4.1 Discretization of the Utility Space

As we mentioned earlier, we assume that the local utility values of the games are in between zero and one. Moreover, $\nabla_i(\bar{q}_{-i})_\alpha$'s are all in between zero and one for any $\bar{q}_{-i}$. For each resource α and a symmetric marginal vector $\bar{q}$, we are going to project $\nabla_i(\bar{q}_{-i})_\alpha$ into a value in $V = \{v_i, i = 0,1,2,..., \lceil\frac{4m}{\epsilon}\rceil \mid v_i = \frac{i\epsilon}{4m}\}$ for some $\epsilon \in (0,1]$. In particular, we let $proj(u) = v$ to be the projection of some value u to a value $v \in V$ such that $u \in (v - \frac{\epsilon}{4m}, v + \frac{\epsilon}{4m})$. Clearly, under the discretization, the error bound from Lemma 2 is going to change by some factor of ϵ.

Proposition 4. *Given the symmetric marginal vector q defined in Proposition 2, and let $proj(u) = v$ be the projection of some value u to a value $v \in V$ such that $u \in (v - \frac{\epsilon}{4m}, v + \frac{\epsilon}{4m})$. We have $\sum_{\alpha\in\mathcal{A}} q_\alpha proj(\nabla(q_{-i})_\alpha) \geq \sum_{\alpha\in\mathcal{A}} q'_\alpha proj(\nabla(q_{-i})_\alpha) - \frac{\epsilon}{2} - O(\delta dnm)$, for any $q' \in [0,1]^m$ that satisfies $Dq' \leq f$ and d is the maximum in-degree of the resource graph G.*

The above proposition follows from the proof of Claim 6 of [4] and our Proposition 2.

3.4.2 Verification of ϵ-MSNE

As discussed in Section 3.2, we can use the primal and dual solutions of the LPs to check for NE. Under our symmetric setting and the projection function, we can rewrite the LPs in Section 3.2 as

Primal

$$\begin{array}{ll} \text{maximize} & {q^*}^T proj(\nabla(q_{-i})) \\ \text{subject to} & Dq^* \leq f \\ & q^*_\alpha \geq 0 \text{ for } \alpha \in \mathcal{A} \end{array}$$

Dual

$$\begin{array}{ll} \text{minimize} & f^T \lambda^* \\ \text{subject to} & D^T \lambda^* \geq proj(\nabla(q_{-i})) \\ & \lambda^*_j \geq 0 \text{ for } j = 1, 2, ..., l, \end{array}$$

where q_i is given in Proposition 2. Indeed, we can use solution of the above LPs to to verify whether q is an approximate NE.

Lemma 3. *Given the symmetric marginal vector q defined in Proposition 2 and let q^* be a best response to q_{-i}, we have $\sum_{\alpha \in \mathcal{A}} q_\alpha proj(\nabla(q_{-i})_\alpha) \geq \sum_{\alpha \in \mathcal{A}} q^*_\alpha proj(\nabla(q_{-i})_\alpha) - \frac{\epsilon}{2} - O(\delta dnm) = f^T \lambda^* - \frac{\epsilon}{2} - O(\delta dnm)$, for any $q' \in [0, 1]^m$ that satisfies $Dq' \leq f$ and d is the maximum in-degree of the resource graph G.*

Furthermore, the above lemma states that as long as we know the value of $f^T \lambda^*$ (the solution of the dual) under the projection function, we can check for an approximate NE. The following lemma characterizes the possible values of λ^*.

Lemma 4 (Lemma 4 of [4]). *The value of $\lambda^* = (\lambda^*_i)_{i=1,2,...,l}$ is such that $\lambda^*_i = c_i v_i$ for some integer constant $0 \leq c_i \leq m$ and $v_i \in V$.*

The above lemma allows us to search for a $\lambda^*_{\bar{q}}$ that corresponds to some marginal vector $\bar{q}$ satisfying the conditions in Lemma 3.

3.4.3 Message Passing Algorithms

To search for a marginal representation of ϵ-MSNE, we propose a two-pass message passing algorithm that runs on a tree decomposition of the constraint moralized resource graph, which is a graph constructed based on the resource graph and the constraints. Our algorithm is a fully polynomial approximation scheme (FPTAS) when the constraint moralized resource graph has bounded treewidth. We introduce and define the concepts below.

Moralized Resource Graph, Constraint Moralized Resource Graph, and Tree Decomposition Recall that a resource graph $G = (\mathcal{A}, E)$ is a directed graph on the resources. Let $C = \{1, ..., l\}$ be the set of constraints, corresponding to rows of the constraint matrix D. We use the notation $D_{\overline{C},\overline{\mathcal{A}}}$ to denote the subcomponent of D consisting of the rows in $\overline{C} \subseteq C$ and columns in $\overline{\mathcal{A}} \subseteq \mathcal{A}$. If $|\overline{C}| = 1$ and $|\overline{\mathcal{A}}| = 1$, then $D_{\overline{C},\overline{\mathcal{A}}}$ is the value corresponds to the row and column.

Moralized graphs (also known as primal graphs) is a standard tool used in the study of many graph-related problems, including Bayesian network inference and constraint satisfaction.

Definition 2. *Given a resource graph $G = (\mathcal{A}, E)$, its moralized resource graph is an undirected graph $MG = (\mathcal{A}, \overline{E} \cup E')$ where $\overline{E}$ is an undirected edge set of E, and $E' = \{\{\alpha, \alpha'\} \in \mathcal{A} \times \mathcal{A} \mid \exists \bar{\alpha} \in \mathcal{A} \text{ s.t. } (\alpha, \bar{\alpha}), (\alpha', \bar{\alpha}) \in E\}$ is a set of undirected edges connecting the parents of their children.*

Given the moralized resource graph, we construct a graph between the resources and the constraints.

Definition 3. *The constraint moralized resource graph is an undirected graph $CMG = (\mathcal{A} \cup C, \overline{E} \cup E' \cup E'')$ where $E'' = \{\{c, \alpha\} \in C \times \mathcal{A} \mid D_{c\alpha} \neq 0\}$ is a set of undirected edges connecting resources and constraints.*

In other words, there is an edge from the constraint to all of the resources involved in the constraint.

We perform tree decomposition on the constraint moralized resource graph. For ease of reference, we provide the definition below.

Table 1: Variables (Var.) and Discretization

Var.	Discretization
q_α	$Q = \{q_i, i = 0, ..., O(\frac{\log m}{\delta^2}) \mid q_i = \frac{i}{\lvert Q \rvert}\}$
λ_c	$\Lambda = \{\lambda_i, i = 0, ..., \frac{4m^2}{\epsilon} \mid \lambda_i = \frac{i\epsilon}{4m}\}$
∂S_c	$S_C = \left\{\pm s_i, i = 0, ..., mO(\frac{\log m}{\delta^2}) \mid s_i = \frac{i}{O(\frac{\log m}{\delta^2})}\right\}$
$\partial \lambda_\alpha$	$S_{\mathcal{A}} = \{\pm s_i, i = 0, ..., \frac{4lm^3}{\epsilon} \mid s_i = \frac{i\epsilon}{4m}\}$
∂t	$T = \left\{t_i, i = 0, ..., \frac{4m^2 O(\frac{\log m}{\delta^2})}{\epsilon} \mid t_i = \frac{i\epsilon}{4mO(\frac{\log m}{\delta^2})}\right\}$
∂f	$F = \left\{\pm f_i, i = 0, ..., \frac{4lm^3}{\epsilon} \mid f_i = \frac{i\epsilon}{4m}\right\}$

Definition 4. *A tree decomposition of an undirected graph $G = (V, E)$ is a tree $T = (B, E')$ where $B \subseteq 2^V$ is a collection of subsets of V that satisfies the following: (a) $\bigcup_{X \in B} X = V$, (b) for each $e = \{e_1, e_2\} \in E$, $\exists X \in B$ such that $e_1, e_2 \in X$, and (c) for any X, X', and $X'' \in B$, if X' is on the path of X and X'', then $X \cap X'' \subseteq X'$. The width of the tree decomposition is $max_{X \in B}(|X| - 1)$. The tree-width is the minimum width of all of the tree decompositions.*

We will present an algorithm that runs on the tree decomposition of the constraint moralized resource graph. Before that, we provide some discretization bounds that will be used in the algorithm.

Discretization. As mentioned earlier, our approach is to search for the marginal representation of an ϵ-MSNE. To do this, we provide discretization bounds for the existence of an ϵ-MSNE in the marginal and utility spaces. Here, we provide other discretization spaces that are necessary for our algorithm. Table 1 lists the variables and their uniform discretization that we used in the following discussion.

To begin, recall that we want to find a marginal vector $q = (q_\alpha)_{\alpha \in \mathcal{A}}$ where $q_\alpha \in Q = \{q_i, i = 0, 1, ..., O(\frac{\log m}{\delta^2}) \mid q_i = \frac{i}{O(\frac{\log m}{\delta^2})}\}$ (as in Lemma 2) such that $Dq \leq f$, $D^T \lambda_q^* \geq proj(\nabla(q_{-i}))$, and $q^T proj(\nabla(q_{-i})) \geq f^T \lambda_q^* - \frac{\epsilon}{2} - O(\delta dnm)$ for $\lambda = (\lambda_c)_{c \in C}$ where $\lambda_c \in \{\lambda_i, i = 0, ..., \frac{4m^2}{\epsilon} \mid \lambda_i = \frac{i\epsilon}{4m}\}$ (as in Lemma 4). Clearly, we are not going to try all the possible $q \in Q^{|m|}$ directly. Instead, our approach is to find feasible $(q_\alpha)_{\alpha \in \mathcal{A}'}$ and $(\lambda_c)_{c \in C'}$ for some ordering of $\mathcal{A}' \subseteq \mathcal{A}$ and $C' \subseteq C$ induced by the tree decomposition incrementally. Thus, we need to keep track of the partial sums of (1) $D_{c,\mathcal{A}} q$ for each constraint c, (2) $D^T_{\alpha,C} \lambda_q$ for each resource α, (3) $q^T proj(\nabla(q_{-i}))$, and (4) $f^T \lambda_q$ for a given q and λ_q. Because D is totally unimodular, its entries can only have values of 0, 1, or -1. Thus, the possible values of the partials sums are finite and have the following discretization spaces.

For (1), the partial sum of a constraint c is bounded by f_c. Since D is totally unimodular, the largest partial sum given any q is no more than $\pm m$ (as discussed in Section 2.3) for any constraint c. Therefore, the partial sum of the constraint c, denoted by ∂S_c, is in $S_C = \left\{\pm s_i, i = 0, ..., mO(\frac{\log m}{\delta^2}) \mid s_i = \frac{i}{O(\frac{\log m}{\delta^2})}\right\}$. Such discretization is because the step size of q is $O(\frac{\log m}{\delta^2})$ and the highest and lowest values of the possible sums are $\pm m$.

For (2), since the λ_c's value is some constant times a value from V and the partial sum can add up to at most ml, the partial sum of a resource constraint α (i.e., $D^T_{\alpha,C}$), denoted by $\partial \lambda_\alpha$, is in $\left\{\pm s_i, i = 0, ..., \frac{4lm^2}{\epsilon} \mid s_i = \frac{i\epsilon}{4m}\right\}$, where $|C| = l$ is the number of constraints. However, we will subtract the partial sum by its right-hand side of the inequality when it is appropriate, as a result, we need to expand the discretization so that $\partial \lambda_\alpha \in S_\alpha = \left\{\pm s_i, i = 0, ..., \frac{4lm^3}{\epsilon} \mid s_i = \frac{i\epsilon}{4m}\right\}$.

For (3), since the possible values of a marginal is in Q, the discretized utility is in V, and the highest partial sum is at most m, the partial utility sum, ∂t (which is a product of elements in Q and V), is in $T = \{t_i, i = 0, ..., \frac{4m^2 O(\frac{\log m}{\delta^2})}{\epsilon} \mid t_i = \frac{i\epsilon}{4mO(\frac{\log m}{\delta^2})}\}$.

For (4), notice that the values of f can be at most m and there are at most l constraints, we have that the partial utility sum, ∂f, is in $F = \left\{\pm f_i, i = 0, ..., \frac{4lm^3}{\epsilon} \mid f_i = \frac{i\epsilon}{4m}\right\}$ (see Section 2.3).

Finally, to ensure we have a marginal vector of ϵ-Nash equilibrium as in Lemma 3, we let $\delta = O\left(\frac{\epsilon}{dmn}\right)$.

The Message Passing Algorithm. Given the graph resulted from the tree decomposition of the constraint moralized graph and the discretization spaces, we provide a message passing algorithm that is an FPTAS when the tree decomposition has a bounded tree width and there is a polynomial number of constraints.

To begin, let $T = (B, E)$ be the decomposed tree. It is clear that, for each $X \in B$, the node X consists of a set of resources and a set of constraints, denoted by $\mathcal{A}_X$ and C_X, respectively. From the moralization and the tree decomposition, for each $\alpha \in \mathcal{A}$, there is an $X \in B$ such that $\nu(\alpha) \subseteq A_X$. This allows us to compute the partial sum of the expected utility $q_\alpha \cdot proj(\nabla(q_{\nu(\alpha)})_\alpha)$. Because of that, we preprocess the tree and designate a node of X to compute the partial sum of the expected utility for each resource α exactly once. Of course, the designated node X should contain the $\nu(\alpha)$. We let $\mathcal{A}^*_X \subseteq A_X$ to denote the set of resources where we compute the expected utility at $X \in B$. Such preprocessing can be done using standard depth-first search or breath-first search.

For clarity, when first present the algorithm when the tree T is a line. Then, we show how we can generalize the algorithm to a tree. Without loss of generality, we assume all of the graphs/trees are connected, otherwise we can just form a new tree by joining the connected components together.

The Line Case. Suppose that the resulted tree decomposition is a line, say $L = (B, E)$. We relabel the nodes of L so that $B = \{1, 2, ..., n\}$ where 1 is the leftmost node and n is the right most node. Thus, the edge set $E = \{\{i, i+1\}, i = 1, ..., n-1\}$ is the adjacent neighbors.

At a high level, the message passing algorithm has an upstream pass and a downstream pass. The upstream pass starts passing messages/tables sequentially from 1 to 2, 2 to 3, ..., n-1 to n. After n has received the messages, the downstream pass begins and starts with n passing message to n-1, n-1 to n-2, so on and so forth until 1 received the message/table from 2, sequentially. The idea of the upstream pass is to keep track of the potential ϵ-MSNE, and the goal of downstream pass is to construct a feasible ϵ-MSNE.

Upstream pass. For each node $i \in B$, we construct a set f_i containing tuples of $Q^{|\mathcal{A}_i|} \times S_C^{|C_i|} \times \Lambda^{|C_i|} \times S_{\mathcal{A}}^{|\mathcal{A}_i|} \times T \times F$ based on $M_{i-1\to i}$ which is the message $i-1$ sends to i (which will be discussed below). More specifically,

$$
\begin{aligned}
f_i = \{&(q_\alpha)_{\alpha\in\mathcal{A}_i}, (\partial S_c)_{c\in C_i}, (\lambda_c)_{c\in C_i}, (\partial\lambda_\alpha)_{\alpha\in\mathcal{A}_i}, \partial t, \partial f) \in Q^{|\mathcal{A}_i|} \times S_C^{|C_i|} \times \Lambda^{|C_i|} \times S_{\mathcal{A}}^{|\mathcal{A}_i|} \times T \times F \mid \\
&\left[\partial t = \sum_{\alpha\in\mathcal{A}^*_i} q_\alpha proj(\nabla(q_{\nu(\alpha)})_\alpha) + \overline{\partial t}\right] (1) \\
&\& \left[\forall c \in C_i \cap C_{i+1},\ \partial S_c = D_{c,\mathcal{A}_i\setminus\mathcal{A}_{i+1}} q_{\mathcal{A}_i\setminus\mathcal{A}_{i+1}} + 1_{[c\in C_{i-1}]}\overline{\partial S_c}\right] (2) \\
&\& \left[\forall c \in C_i \setminus C_{i+1},\ \partial S_c = D_{c,\mathcal{A}_i} q_{\mathcal{A}_i} + 1_{[c\in C_{i-1}]}\overline{\partial S_c}\right] (3) \\
&\& \left[\forall \alpha \in (\mathcal{A}_i \setminus \mathcal{A}^*_i) \cap \mathcal{A}_{i+1},\ \partial\lambda_\alpha = D^T_{\alpha, C_i\setminus C_{i+1}} \lambda_{C_i\setminus C_{i+1}} + 1_{[\alpha\in A_{i-1}]}\overline{\partial\lambda_\alpha}\right] (4) \\
&\& \left[\forall \alpha \in (\mathcal{A}_i \setminus \mathcal{A}^*_i) \setminus \mathcal{A}_{i+1},\ \partial\lambda_\alpha = D^T_{\alpha, C_i} \lambda_{C_i} + 1_{[\alpha\in A_{i-1}]}\overline{\partial\lambda_\alpha}\right] (5) \\
&\& \left[\forall \alpha \in \mathcal{A}^*_i \cap \mathcal{A}_{i+1},\ \partial\lambda_\alpha = D^T_{\alpha, C_i\setminus C_{i+1}} \lambda_{C_i\setminus C_{i+1}} + 1_{[\alpha\in A_{i-1}]}\overline{\partial\lambda_\alpha} - proj(\nabla(q_{\nu(\alpha)})_\alpha)\right] (6) \\
&\& \left[\forall \alpha \in \mathcal{A}^*_i \setminus \mathcal{A}_{i+1},\ \partial\lambda_\alpha = D^T_{\alpha, C_i} \lambda_{C_i} + 1_{[\alpha\in A_{i-1}]}\overline{\partial\lambda_\alpha} - proj(\nabla(q_{\nu(\alpha)})_\alpha)\right] (7) \\
&\& \left[\partial f = f^T_{C_i\setminus C_{i+1}} \lambda_{C_i\setminus C_{i+1}} + \overline{\partial f}\right] (8) \\
&\& [((q_\alpha)_{\alpha\in\mathcal{A}_i\cap\mathcal{A}_{i-1}}, (\overline{\partial S_c})_{c\in C_i\cap C_{i-1}}, (\lambda_c)_{c\in C_i\cap C_{i-1}}, (\overline{\partial\lambda_\alpha})_{\alpha\in\mathcal{A}_i\cap\mathcal{A}_{i-1}}, \overline{\partial t}, \overline{\partial f}) \in M_{i-1\to i}](9)\},
\end{aligned}
$$

where & denotes the regular "and" condition.

The (1) condition keeps track of the partial sums of the expected utilities of the resources in $\mathcal{A}^*_j$, $j = 1$ to i-1, and add the expected utilities of resources in $\mathcal{A}^*_i$. The (2) and (3) conditions accumulate the partial sum of the constraints in C_i (i.e., we are keeping track of the C_i constraints of D to ensure $D_{C_i,\mathcal{A}_i}\, q_{\mathcal{A}_i} \leq f_{C_i}$). This boils down to two cases given a $c \in C_i$, either c appears in C_{i+1} (condition (2)) or c does not appear in C_{i+1} (condition (3)). If c appears in C_{i+1} (condition (2)), then we keep track of the partial sum of the resources together with c that will not appear in C_{i+1}, otherwise we will just include all of the resources in C_i if c doesn't appear in C_{i+1} (condition (3)) since c

will not appear again in the later nodes (due to the property of the tree decomposition). In either case, we need to the include the (possible) existing partial sum $\overline{\partial S_c}$ for c that appears in the previous $j = i-1, ...$ nodes.

For conditions (4), (5), (6), and (7), they are similar to conditions (2) and (3) but instead, we want to keep track of partial sums of the dual constraints (i.e., $D^T_{\mathcal{A}_i,C_i}\lambda_{C_i} \geq proj(\nabla(q_{\nu(\mathcal{A}_i)})_{\mathcal{A}_i})$). Clearly, we can only compute the projected expected utility for the resources in $\mathcal{A}_i^*$. Therefore, we have cases (6) and (7) for which we can compute the expected utilities and cases (4) and (5) for which we cannot. For (4) and (5), we just want to keep track of the partial constraints of $D_{\alpha,C_i}\lambda_{C_i}$; either we include all of the constraints in C_i in the partial sum (condition (5)) or the constraints that do not appear in C_{i+1}. For (6) and (7), this is similar except that we subtract from the projected expected utilities for α in $\mathcal{A}_i^*$. The reason we are doing this is because we will not able to compute the projected expected utilities after this, therefore we subtract from the partial sum. In all of these cases, we need to include the existing partial sum of α.

For condition (8), we keep track of the partial sum of $f^T_{C_i}\lambda_{C_i}$. We only need to add the partial sum for those constraints that will not appear again in the tree. Finally (9) ensures that our tuples are consistent with the messages that $i-1$ send to i. We observe that the resulting set f_i has cardinality bounded by $poly(n, m, l, 1/\epsilon)$, if the width of the tree decomposition is bounded by a constant.

Message Table. Given the set of tuples f_i, we are ready to discuss the message $M_{i\to i+1}$ i sends to $i+1$. We first define and construct an auxiliary set M_{f_i} that will facilitate the constructing of the message and also make the downstream pass more clear.

$$\begin{aligned}M_{f_i} = \{&(q_\alpha)_{\alpha\in\mathcal{A}_i}, (\partial S_c)_{c\in C_i}, (\lambda_c)_{c\in C_i}, (\partial\lambda_\alpha)_{\alpha\in\mathcal{A}_i}, \partial t, \partial f) \in f_i \mid \\ &[\forall c \in C_i \setminus C_{i+1},\ \partial S_c \leq f_c] \,\&\, [\forall \alpha \in \mathcal{A}_i \setminus \mathcal{A}_{i+1},\ \partial\lambda_\alpha \geq 0] \,\&[(i=n) \implies \partial t \geq \partial f - \epsilon]\},\end{aligned}$$

and the message from i to $i+1$ is

$$\begin{aligned}M_{i\to i+1} = \{&((q_\alpha)_{\mathcal{A}_i\cap\mathcal{A}_{i+1}}, (\partial S_c)_{c\in C_i\cap C_{i+1}}, (\lambda_c)_{c\in C_i\cap C_{i+1}}, (\partial\lambda_\alpha)_{\alpha\in\mathcal{A}_i\cap\mathcal{A}_{i+1}}, \partial t, \partial f) \mid \\ &\exists((q_\alpha)_{\alpha\in\mathcal{A}_i}, (\partial S_c)_{c\in C_i}, (\lambda_c)_{c\in C_i}, (\partial\lambda_\alpha)_{\alpha\in\mathcal{A}_i}, \partial t, \partial f) \in M_{f_i}\}.\end{aligned}$$

When $i = 0$, $C_0 = A_0 = C_{n+1} = A_{n+1} = \emptyset$ and $M_{0\to 1} = \{\emptyset, \emptyset, \emptyset, \emptyset, 0, 0\}$. When $i = n$, $C_{n+1} = A_{n+1} = \emptyset$ and n does not send message from n to $n+1$ but still construct the table M_{f_n}.

Downstream Pass. Now that we have completed the upstream pass, we have enough information to construct an ϵ-MSNE. We will do this by starting selecting partial ϵ-MSNE for n, n-1, ..., 1, sequentially. In particular, given the selected partial ϵ-MSNE at node i, i will send node $i-1$ the feasible strategies that $i-1$ should select to ensure ϵ-MSNE. In particular, i will construct a return feasible set

$$\begin{aligned}R_i = \{&((q_\alpha)_{\alpha\in\mathcal{A}_i}, (\partial S_c)_{c\in C_i\setminus C_{i+1}}, (\overline{\partial S_c})_{c\in C_i\cap C_{i+1}}, (\lambda_c)_{c\in C_i}, (\partial\lambda_\alpha)_{\alpha\in\mathcal{A}_i\setminus\mathcal{A}_{i+1}}, (\overline{\partial\lambda_\alpha})_{\alpha\in\mathcal{A}_i\cap\mathcal{A}_{i+1}}, \overline{\partial t}, \overline{\partial f}) \in M_{f_i} \mid \\ &((q_\alpha)_{\alpha\in\mathcal{A}_i\cap\mathcal{A}_{i+1}}, (\overline{\partial S_c})_{c\in C_i\cap C_{i+1}}, (\lambda_c)_{c\in C_i\cap C_{i+1}}, (\overline{\partial\lambda_\alpha})_{\alpha\in\mathcal{A}_i\cap\mathcal{A}_{i+1}}, \overline{\partial t}, \overline{\partial f}) \\ &\in R_{i+1\to i}[(q_\alpha)_{\alpha\in\mathcal{A}_{i+1}}, (\partial S_c)_{c\in C_{i+1}}, (\lambda_c)_{c\in C_{i+1}}, (\partial\lambda_\alpha)_{\alpha\in\mathcal{A}_{i+1}}, \partial t, \partial f]\},\end{aligned}$$

and after selecting a tuple $((q_\alpha)_{\alpha\in\mathcal{A}_i}, (\partial S_c)_{c\in C_i}, (\lambda_c)_{c\in C_i}, (\partial\lambda_\alpha)_{\alpha\in\mathcal{A}_i}, \partial t, \partial f) \in R_i$, i send the following to $i-1$

$$\begin{aligned}&R_{i\to i-1}[(q_\alpha)_{\alpha\in\mathcal{A}_i}, (\partial S_c)_{c\in C_i}, (\lambda_c)_{c\in C_i}, (\partial\lambda_\alpha)_{\alpha\in\mathcal{A}_i}, \partial t, \partial f] = \\ &\{((q_\alpha)_{\mathcal{A}_i\cap\mathcal{A}_{i-1}}, (\overline{\partial S_c})_{c\in C_i\cap C_{i-1}}, (\lambda_c)_{c\in C_i\cap C_{i-1}}, (\overline{\partial\lambda_\alpha})_{\alpha\in\mathcal{A}_i\cap\mathcal{A}_{i-1}}, \overline{\partial t}, \overline{\partial f}) \in M_{i-1\to i} \mid \\ &\text{ the tuples satisfy conditions (1) - (8)}\}.\end{aligned}$$

When $i = n$, $R_n = M_{f_n}$. At each node i, i constructs R_i based on message from $i+1$, selects a tuple from R_i, and send messages to $i-1$ based on the selected tuple.

Generalization to Trees. Our message-passing algorithm can be generalized easily to trees. It has been shown that given a tree decomposition, we can modify it in polynomial time to a *nice tree decomposition* with the same treewidth and a polynomial increase in the number of nodes, where each node of the nice tree has at most two children (say left and right). Thus, we can construct the sets f_i, M_{f_i}, $M_{i\to parent}$, R_i, $R_{i\to left}$, and $R_{i\to right}$ based on the messages from the two children, with only polynomial increase in the computation and space requirements. For the upstream pass, a node i with a left child and a right child will construct f_i based on the messages from its left child and right child. The equations (1)-(8) will include partial sums from the left and right children (as opposed to just based on $i-1$) and $i+1$ will be replaced by the parent of i. The tables M_{f_i} and $M_{i \to parent}$ will be constructed the same way as before by replacing the index $i+1$ with the parent of i. For the downstream pass, the R_i can be constructed similarly using the message from the parent of i. The messages $R_{i \to left}$ and $R_{i \to right}$ can be constructed the same way but based on the messages $M_{left \to i}$ and $M_{right \to i}$ that satisfy the conditions (1)-(8).

Theorem 1. *There is an FPTAS for computing MSNE in RGGs whose constraint moralized resource graphs have bounded tree width. In particular, the algorithm finds an ϵ-MSNE in time poly$(n, m, l, \frac{1}{\epsilon})$.*

Proof. (Again, we present the proof for the line case for notational clarity. The arguments for the tree case is the same.) To prove that our message-passing algorithm is an FPTAS, we need to show the downstream pass/assignment provides an ϵ-MSNE and the algorithm runs in poly$(n, m, l, \frac{1}{\epsilon})$.

We begin to show the selected tuples of the nodes form an ϵ-MSNE. We let

$$((q^i_\alpha)_{\alpha \in \mathcal{A}_i}, (\partial S^i_c)_{c \in C_i}, (\lambda^i_c)_{c \in C_i}, (\partial \lambda^i_\alpha)_{\alpha \in \mathcal{A}_i}, \partial t^i, \partial f^i)$$

to be the selected tuples by each node i. For clarity, we use the superscript i to specify the selected tuple of node i. By construction, for each node i, if $\alpha \in A_i \cap A_{i-1}$ and $c \in C_i \cap C_{i-1}$, $q^i_\alpha = q^{i-1}_\alpha$ and $\lambda^i_c = \lambda^{i-1}_c$, respectively. This construction allows us to maintain consistency in the assignment.

Given these tuples, we need to show that (1) $Dq \leq f$, (2) $D^T\lambda \geq proj(\nabla(q_{-i}))$, and (3) $\sum_{\alpha \in \mathcal{A}} q_\alpha \cdot proj(\nabla(q_{-i})_\alpha) \geq f^T\lambda - \epsilon$.

(1) For each constraint $c \in C$, let $s_c \in [n]$ and $t_c \in [n]$ ($s_c \leq t_c$) be the indices of the nodes in which c appears first and last in the tree decomposition, respectively. From the properties of the tree decomposition, c must appear in the nodes of the paths from X_{s_c} to X_{t_c}. Thus, we have

$$\begin{aligned} f_c \geq \partial S^{t_c}_c &= D_{c,\mathcal{A}_{t_c}} q^{t_c}_{\mathcal{A}_{t_c}} + 1_{[c \in C_{t_c - 1}]} \partial S^{t_c - 1}_c \\ &= D_{c,\mathcal{A}_{t_c}} q^{t_c}_{\mathcal{A}_{t_c}} + \sum_{i=s_c}^{t_c - 1} D_{c, \mathcal{A}_i \setminus \mathcal{A}_{i+1}} q^i_{\mathcal{A}_i \setminus \mathcal{A}_{i+1}} = D_c q, \end{aligned}$$

where the first inequality is by construction in which t_c selects a tuple with a feasible $f_c \geq \partial S^{t_c}_c$ from R_{t_c}, the first equality is from our construction where $\partial S^{t_c}_c$ is its current sum plus the previous partial sum, the second equality is from decomposing each partial sum term, and the last equality is by the property of the tree decomposition of our moralized constraint graph where the resources (and their $q'_\alpha s$) in the constraint c must be in one of the nodes from the path of s_c to t_c.

(2) We can use a similar arguments as in (1) to show that λ is feasible. For each resource $\alpha \in \mathcal{A}$, let $s_\alpha \in [n]$, $a^*_\alpha \in [n]$, and $t_\alpha \in [n]$ to be the indices of the nodes in which α appears first, designated to compute its expected utility, and last in the tree decomposition, respectively. It is not hard to see that $s_\alpha \leq a^*_\alpha \leq t_\alpha$ and α appears in the path from X_{s_α} to X_{t_α}. Thus, we have

$$\begin{aligned} &0 \leq \partial \lambda^{t_\alpha}_\alpha = D^T_{\alpha, C_{t_\alpha}} \lambda^{t_\alpha}_{C_{t_\alpha}} + 1_{[\alpha \in A_{t_\alpha - 1}]} \partial \lambda^{t_\alpha - 1}_\alpha - 1_{[a^*_\alpha = t_\alpha]} q^{a^*_\alpha}_\alpha proj(\nabla(q^{a^*_\alpha}_{\nu(\alpha)})_\alpha \\ &= \sum_{i=a^*_\alpha+1}^{t_\alpha - 1} D^T_{\alpha, C_i \setminus C_{i+1}} \lambda^i_{C_i \setminus C_{i+1}} + 1_{[a^*_\alpha \neq t_\alpha]} D^T_{\alpha, C_{a^*_\alpha} \setminus C_{a^*_\alpha + 1}} \lambda^{a^*_\alpha}_{C_{a^*_\alpha} \setminus C_{a^*_\alpha+1}} - q^{a^*_\alpha}_\alpha proj(\nabla(q^{a^*_\alpha}_{\nu(\alpha)})_\alpha) \\ &\qquad\qquad + \sum_{i=s_c}^{a^*_\alpha - 1} D^T_{\alpha, C_i \setminus C_{i+1}} \lambda^i_{C_i \setminus C_{i+1}} + D^T_{\alpha, C_{t_\alpha}} \lambda^{t_\alpha}_{C_{t_\alpha}} \\ &= \sum_{i=s_\alpha}^{t_\alpha - 1} 1_{[i \neq a^*_\alpha \text{ or } i = a^*_\alpha \neq t_\alpha]} D^T_{\alpha, C_i \setminus C_{i+1}} \lambda^i_{C_i \setminus C_{i+1}} + D^T_{\alpha, C_{t_\alpha}} \lambda^{t_\alpha}_{C_{t_\alpha}} - q^{a^*_\alpha}_\alpha proj(\nabla(q^{a^*_\alpha}_{\nu(\alpha)})_\alpha) = D^T_\alpha \lambda - q^{a^*_\alpha}_\alpha proj(\nabla(q^{a^*_\alpha}_{\nu(\alpha)})_\alpha), \end{aligned}$$

where the first inequality is by construction in which t_α selects a tuple with a feasible $0 \leq \partial \lambda^{t_\alpha}_\alpha$ from R_{t_α}, the first equality is from decomposing each partial sum term, the second equality is from combining the sums, and the third equality is by the property of the tree decomposition of our moralized constraint graph where the constraints (and their $\lambda_c' s$) in the resource α must be in one of the nodes from the path of s_α to t_α. If $a^*_\alpha = t_\alpha$ or $a^*_\alpha = s_\alpha$, then the corresponding sum is zero.

Now, we need to argue (3). From R_n, we select a tuple such that $\partial t^n \geq \partial f^n - \epsilon$. Expanding,

$$\partial t^n = \sum_{\alpha \in \mathcal{A}^*_n} q_\alpha proj(\nabla(q_{\nu(\alpha)})_\alpha) + \partial t^{n-1} = \sum_{\alpha \in \mathcal{A}} q_\alpha proj(\nabla(q_{\nu(\alpha)})_\alpha),$$

$$\partial f^n = f^T_{C_n} \lambda^n_{C_n} + \partial f^{n-1} = \sum_{i=1}^{n-1} f^T_{C_i \backslash C_{i+1}} \lambda^i_{C_i \backslash C_{i+1}} + f^T_{C_n} \lambda_{C_n} = f^T \lambda,$$

which proves (3).

Finally, the running time of the algorithm can be reasoned as follows. The size of f_i, R_i, $M_{i \to i+1}$, and $R_{i \to i-1}$ is at most poly$(n, m, l, \frac{1}{\epsilon})$ when the treewidth is bounded, and it takes poly$(n, m, l, \frac{1}{\epsilon})$ time to construct. $\square$

3.4.4 RGGs with Constant Number of Player Types

So far, we have been discussing the case where the players have the same (single) type (i.e., polytope). In fact, we can generalize our message passing algorithms to cases where the players have a constant number of types.

Definition 5. *An RGG, $\Gamma = (N, \mathcal{A}, \{S_i\}_{i=1}^n, G, \{u^\alpha\}_{\alpha \in \mathcal{A}})$, is k-symmetric if and only if the players are partitioned into k classes, and for any player i in class $j \in \{1, ..., k\}$, $D_i = D^j$, $f_i = f^j$ for some $D^j \in \mathbb{Z}^{l_i^j \times |\mathcal{A}|}$ and $f^j \in \mathbb{Z}^{l_i^j}$ (and therefore $S_i = S^j$, and $P_i = P^j$).*

Proposition 5. *For every k-symmetric multilinear RGGs, there is a k-symmetric Nash equilibrium in which every player in the same class plays the same strategy.*

It can be verified our results (Lemmas 2, 3, and 4) and tools (constrained moralized resource graph and message-passing algorithm) for computing symmetric approximate MSNE in symmetric multilinear RGGs can be used to compute k-symmetric approximate MSNE in k-symmetric multilinear RGGs by considering k different symmetric strategies for the k classes of players.

More specifically, we construct the constrained moralized resource graph from the resources and the constraints from the polytopes of the k-classes and perform a tree decomposition (with bounded treewidth) on the resulting graph. In the construction of tables/sets for each node i in the tree decomposition, we need to keep track of $(q^j_\alpha)_{\alpha \in \mathcal{A}_i}$, $(\partial S_c)_{c \in C^j_i}$, $(\lambda_c)_{c \in C^j_i}$, $(\partial \lambda^j_\alpha)_{\alpha \in \mathcal{A}_i}$, ∂t^j, and ∂f^j for each class $j = 1, ..., k$. This results in running time and table size exponential in k and the treewidth.

Proposition 6. *There is an FPTAS for computing a k-symmetric Nash equilibrium in a k-symmetric RGG whose constraint moralized resource graph has constant treewidth, when k is a constant.*

4 Applications to Specific Game Classes

Many classes of games studied previously can be captured and modeled using mulitlinear RGGs. These include security games, congestion games, and bilinear games. Below we will show how our algorithmic results can be applied to compute approximate Nash equilibrium in these games.

4.1 Security Games

In a typical security game [29] between a defender (d) and an attacker (a), there is a set T of targets, the defender can only protect m of the targets and attacker can attack one target. The defender and attacker's utilities depend on whether the attacked target is covered. One thus only need to store 4 numbers per target, for both players' utilities when that target is attacked and covered and when it is attacked and not 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 $S_a = \{x \in \{0,1\}^{3|T|} \mid \sum_{a_t} x_{a_t} = 1, x_{b_{0t}} = x_{b_{1t}} = 0 \forall t \in T\}$. For the defender,

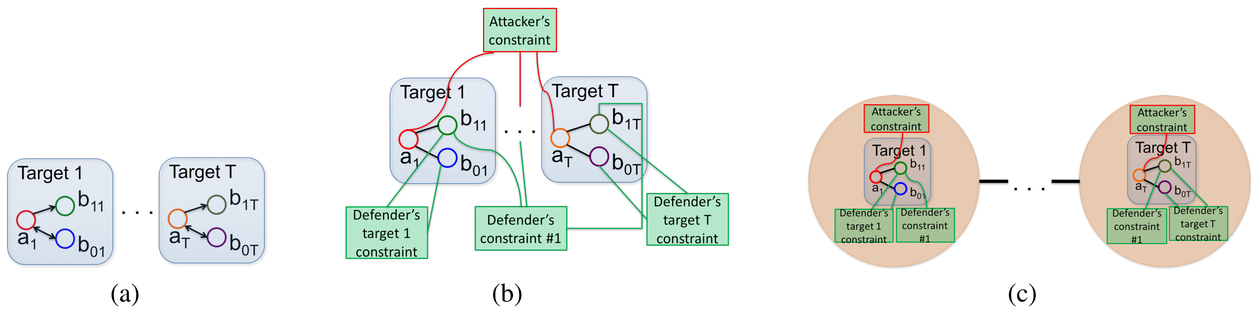

Figure 1: T**-target Security Games**. (a) RGG representation of Security Games, (b) Constraint Moralized Resource Graph, and (c) a tree decomposition of the Constraint Moralized Resource Graph. Note that the unary constraints are not shown in (b) and (c)

we construct a constraint matrix D_b of size $2|T|+1$ 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 $d_t = (d_{t\alpha})_{\alpha\in\mathcal{A}}$ such that $d_{t\alpha} = 1$ only at b_{0t} and b_{1t} and a row of $d_t = (d_{t\alpha})_{\alpha\in\mathcal{A}}$ such that $d_{t\alpha} = 1$ only at a_t. Thus, the strategy set of a defender is $S_b = \{x \in \{0,1\}^{3|T|} \mid D_b x = (m, 1, ..., 1, 0, ..., 0)^T\}$. 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(a) 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 $t \in T$. In particular, given a strategy of the attacker $s_a \in S_a$ and a strategy of the defender $s_d \in S_d$, $u_a(s_a, s_d) = \sum_{a_t} s_{a,a_t} u^{a_t}(s_{d,b_{0t}})$ and $u_d(s_d, s_a) = \sum_{b_{0t}} s_{d,b_{0t}} u^{b_{0t}}(s_{a,a_t}) + \sum_{b_{1t}} s_{d,b_{1t}} s_{d,b_{1t}} u^{b_{1t}}(s_{a,a_t})$. It is not hard to see that the constraint matrices of the attacker and the defender are totally unimodular, and the strategies of the players satisfy condition (b) of Proposition 1.

Given the RGG representation of the security game, we can proceed to construct its moralized constraint resource graph, as shown in Figure 1(b). We then can perform a tree decomposition of the moralized constraint resource graph. We provide a tree decomposition in Figure 1(c). For any number of targets, each node of this tree decomposition has six nodes of the moralized constraint resource graph plus the three missing unary constraint nodes, and therefore has width eight. Since we have only two classes of players (an attacker and a defender) and constant treewidth, our message passing algorithm yields an FPTAS to computing an approximate Nash equilibrium in this games.

Moreover, our result holds for more general classes of security games in which the attacker can have any number of attacks, arbitrary strategy constraints for the attacker and defender as long as their constraint matrices are totally unimodular and the treewidth of the moralized constraint resource graph is bounded.

Proposition 7. *Given a security game, if the strategy constraint matrices for the attacker and defender are totally unimodular, and treewidth of the moralized constraint resource graph is bounded, there is an FPTAS for computing MSNE.*

Example 1. *Consider a security game as defined above, but with a more complex attacker strategy space. In particular the targets correspond to edges in a network with a source and a destination, and the attacker can choose a path from source to destination, thereby attacking all edges on the path chosen. It is known that the set of valid paths can be encoded compactly as linear constraints with a totally unimodular constraint matrix. If the network has constant treewidth, the constraint moralized resource graph also has constant treewidth, and Proposition 7 applies.*

4.2 Congestion Games

Congestion games model the situation in which there is a set of resources, and each player selects a subset of resources. The cost of using a resource depends on the total number of agents using it. As such, the agent's goal is to select a subset of resources that minimizes the total cost. More specifically, we have a set of player N and a set of resources R. The pure-strategy set of player i is S_i, and each $s_i \in S_i$ is represented by a binary vector of size $|R|$.

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. For each player i, $S_i = P_i \cap \{0,1\}^{|R|}$ where $P_i = \{x \in \mathbb{R}^{|R|} \mid D_i x \leq f_i\}$ is the convex hull of the set of pure strategies S_i. The utility u^r of a resource corresponds to the congestion cost function for resource $r \in R$.

When the constraint matrix D_i of P_i of each player i is totally unimodular, then we have totally unimodular congestion games [4, 10], which have been introduced and studied recently. The totally unimodular congestion games include network congestion games. Indeed, our result applies to this type of congestion games and generalized the results of [4]. In particular, their FPTAS to compute an approximate MSNE for totally unimodular congestion games requires each resource to only be involved in a constant number of constraints and each constraint can only involve a small/bounded number of resources. Our result lifts these restrictions and applies as long as the treewidth of the moralized constraint graph is bounded.

Proposition 8. *Given a totally unimodular congestion game with k player types where the strategy constraint matrices for the k player types are totally unimodular, if the treewidth of the moralized constraint resource graph is bounded, there is an FPTAS for computing k-symmetric MSNE for constant k.*

Example 2 (Network Congestion)**.** *Consider network congestion games [11], where resources correspond to edges in a network, and each player chooses a path from a source to a destination in the network. [4] provided an FPTAS when the network has constant treewidth and constant degree. Our result provides an FPTAS when the network has constant treewidth, without restriction on the degree.*

Example 3 (Security Game with Multiple Attackers)**.** *Consider the following generalization to security games, now with n_a attackers. Each attacker can attack K targets. Payoffs for defender and attackers are sums of contributions from each target, which depend on (a) the number of attackers choosing to attack that target, and (b) whether defender protects that target. The attackers are interchangeable so this is a 2-symmetric game. This game combines the utility structure of congestion games (among the attackers) and security games (between defender and attackers). The RGG representation has resource graph similar to Figure 1(a), except now there are self-loops on each attacker node a_t for each target. It can be verified that Assumption 2 remains satisfied, and Figure 1(c) remains a tree decomposition for the new constraint moralized resource graph. Thus our result applies to yield an FPTAS.*

4.3 Bilinear Games

A bilinear game [12] is played between two players 1 and 2. The strategy spaces of the players are described using compact polytopes where $X = \{x \in \mathbb{R}^M \mid Ex = e, x \geq 0\}$ is player 1's strategy space and $Y = \{y \in \mathbb{R}^N \mid Fy = f, y \geq 0\}$ is player 2's strategy space for some $E \in \mathbb{R}^{k_1 \times M}$ and $e \in \mathbb{R}^{k_1}$ and $F \in \mathbb{R}^{k_2 \times N}$ and $f \in \mathbb{R}^{k_2}$. Player 1 and player 2 have payoff matrices $A \in \mathbb{R}^{M \times N}$ and $B \in \mathbb{R}^{M \times N}$, respectively, describing their payoffs when the players play the actions in X and Y. In particular, given (pure) strategies $x \in X$ and $y \in Y$, $u_1(x, y) = x^T Ay$ and $u_2(y, x) = x^T By$.

It turns out that we can represent a variant of bilinear games using multilinear RGGs and compute approximate MSNE using our message passing algorithm efficiently when the resource graphs of the bilinear games have bounded treewidth.

4.3.1 RGG Representations of a Large Class of Bilinear Games

Notice that in the general definition of bilinear games, the polytopes of the players are not restricted to the extreme points of 0 and 1. Here, we consider the case in which the polytopes of the players are (binary) integer vectors as in the definition of RGGs, and when the matrices E and F are totally unimodular. For the RGG representation, we create the games with $M + N$ resources. In particular, player 1's polytopal strategy space is $S_1 = \{x \in [0,1]^{M+N} \mid \left[\frac{E_{k_1 \times M} | \mathbf{0}_{k_1 \times N}}{\mathbf{0}_{N \times M} | I_{N \times N}}\right] x = [e | \mathbf{0}_{1 \times N}]\} \cap \{0,1\}^{M+N}$, player 2's polytopal strategy space is $S_2 = \{y \in [0,1]^{N+M} \mid \left[\frac{F_{k_2 \times N} | \mathbf{0}_{k_2 \times M}}{\mathbf{0}_{M \times N} | I_{M \times M}}\right] y = [f | \mathbf{0}_{1 \times M}]\} \cap \{0,1\}^{N+M}$ where I is the identity matrix and **0** is the all zero matrix, and these constraint matrices are totally unimodular if E and F are totally unimodular. Let $R = \{r_1, ..., r_M\}$ and $C = \{c_1, ..., c_N\}$ denote the columns and the rows of the payoff matrices A and B, respectively.

The resource graph is a bipartite graph $G = (R \cup C, E_A \cup E_B)$ where $E_A = \{(c, r) \in C \times R \mid A_{c,r} \neq 0\}$ and $E_B = \{(r, c) \in R \times C \mid B_{r,c} \neq 0\}$ are edge sets specifying the payoff contributions of the entries in the matrices. For each resource $r_i \in R$ and any strategy $s_2 \in S_2$, $u^{r_i}(c^{r_i}) = \sum_{c_j \in Pa(r_i)} s_{2,c_j} A_{i,j}$, and for each resource $c_i \in C$ and any strategy $s_1 \in S_1$, $u^{c_i}(c^{c_i}) = \sum_{r_j \in Pa(c_i)} s_{1,r_j} B_{j,i}$. It is also easy to see that the RGG representation satisfies condition 1(b) of Proposition 1.

Clearly, if we follow our earlier step and construct the constraint moralized graph, we can apply our algorithm directly if the graph has bounded treewidth.

Proposition 9. *Given a bilinear game, if the strategy constraint matrices for both players are totally unimodular, and treewidth of the moralized constraint resource graph is bounded, there is an FPTAS for computing MSNE.*

In fact, we can do better for bilinear games, and we do not need to moralize the resource graph at all. Moreover, we can perform a tree decomposition on a variant of the (undirected) resource graph and obtain an FPTAS to compute MSNE.

Theorem 2. *Given a bilinear game, if the strategy constraint matrices for both players are totally unimodular, and treewidth of the constraint resource graph (without moralization) is bounded, there is an FPTAS for computing MSNE.*

Proof. (Sketch) First, we define a different but utility equivalence RGGs $\overline{G}$ of the original RGGs G with more constraints and resources. In particular, $\overline{G} = (R \cup C \cup \overline{R} \cup \overline{C}, \overline{E}_A \cup \overline{E}_B)$ where $\overline{R} = \{r_{ij}, j \in Pa(r_i) \mid \forall r_i \in R\}$, $\overline{C} = \{c_{ij}, j \in Pa(c_i) \mid \forall c_i \in C\}$, $\overline{E}_A = \{(c_i, r_{jc_i}) \mid \forall (c_i, r_j) \in E_A\}$, and $\overline{E}_B = \{(r_i, c_{jr_i}) \mid \forall (r_i, c_j) \in E_A\}$ where $Pa(i)$ is the set of parents of node i under the graph G. We need to extend the strategies of the players to include the added resources $\overline{R}$ and $\overline{C}$. In particular, to construct the polytopes for the new strategies $\overline{S}_1$ and $\overline{S}_2$, we need to (1) extend the size of dimensional of the polytopes S_1 and S_2 to $|R| + |C| + |\overline{R}| + |\overline{C}| \leq M + N + d_{max}(M + N) = (d_{max} + 1)(M + N)$ with bounded degree d_{max} of the original graph, (2) add in the unary constraints to $\overline{S}_1$ and $\overline{S}_2$ so that player 1 and player 2 do not use the resources in $\overline{C}$ and $\overline{R}$, respectively, and (3) add (binary) constraints to ensure that, for each $r_i \in R$, r_i is used together with each $r_{ic_j} \in \overline{R}$ for $c_j \in Pa(r_i)$ and for each $c_i \in C$, c_i is used together with each $c_{ir_j} \in \overline{C}$ for $r_j \in Pa(c_i)$. For resource $\alpha \in R \cup C$ and for any strategy, $\overline{u}^{\alpha} = 0$ since α does not depend on any other resources directly. For resource $r_{ic_j} \in \overline{R}$ and $\overline{s}_2 \in \overline{S}_2$, $\overline{u}^{r_{ic_j}}(\overline{s}_{2,c_j}) = \overline{s}_{2,c_j} A_{i,j}$. For resource $c_{ir_j} \in \overline{C}$ and $\overline{s}_1 \in \overline{S}_1$, $\overline{u}^{c_{ir_j}}(\overline{s}_{1,r_j}) = \overline{s}_{1,r_j} B_{j,i}$.

Next, we show that we can map any strategy of the original game to the strategy of the new games and vice versa such that they have the same payoffs. Given $s_1 \in S_1$ and $s_2 \in S_2$, we construct $\overline{s}_1 \in \overline{S}_1$ so that $\overline{s}_{1r_i} = 1$ if $s_{1r_i} = 1$ for all $r_i \in R$ and $\overline{s_{1r_{ic_j}}} = 1$ if $s_{1r_i} = 1$ for all $r_{ic_j} \in \overline{R}$ and $c_j \in Pa(r_i)$ and $\overline{s}_{2c_i} = 1$ if $s_{2c_i} = 1$ for all $c_i \in C$ and $\overline{s}_{2c_{ir_j}} = 1$ if $s_{2c_i} = 1$ for all $c_{ir_j} \in \overline{C}$ and $r_j \in Pa(c_i)$ (all other entries are zero). The utility of player 1 is

$$\begin{aligned}
\overline{u}_1(\overline{s}_1, \overline{s}_2) &= \sum_{r_i \in R} (\overline{s}_{1r_i} \overline{u}^{r_i} + \sum_{c_j \in Pa(r_i)} \overline{s}_{1r_{ic_j}} \overline{u}^{r_{ic_j}}(c^{r_{ic_j}})) \\
&= \sum_{r_i \in R} (\overline{s}_{1r_i} \overline{u}^{r_i} + \sum_{c_j \in Pa(r_i)} \overline{s}_{1r_{ic_j}} \overline{s}_{2,c_j} A_{i,j}) \\
&= \sum_{r_i \in R} \overline{s}_{1r_i} \sum_{c_j \in Pa(r_i)} \overline{s}_{1r_{ic_j}} \overline{s}_{2,c_j} A_{i,j} \\
&= \sum_{r_i \in R} \overline{s}_{1r_i} \sum_{c_j \in Pa(r_i)} \overline{s}_{2,c_j} A_{i,j},
\end{aligned}$$

which obtains the same values as $u_1(s_1, s_2) = \sum_{r_i \in R} s_{1r_i} u^{r_i}(c^{r_i}) = \sum_{r_i \in R} s_{1r_i} \sum_{c_j \in Pa(r_i)} s_{2,c_j} A_{i,j}$. This is similar for $\overline{u}_2$.

On the other hand, given $\overline{s}_1 \in \overline{S}_1$ and $\overline{s}_2 \in \overline{S}_2$, we can construct $s_1 \in S_1$ and $s_2 \in S_2$ by setting the corresponding resources in $R \cup C$ to be the same values as those in $\overline{s}_1$ and $\overline{s}_2$. The payoff-equivalence can be shown easily.

Next, we perform tree decomposition on the constraint graph of the original RGG and show that if the tree decomposition of the original RGG is bounded, we can use it to construct a bounded tree decomposition of the new RGG. Let $T = (B, E)$ be the (nice) tree decomposition of the original RGG. The new tree composition $\overline{T} = (\overline{B}, \overline{E})$ is a copy of T where each node $\overline{X} \in \overline{B}$ is a union of $X \in B$ and some nodes in $\overline{R}$ and $\overline{C}$. In particular, to obtain $\overline{X}$, we add all of the elements in $X \in B$ to $\overline{X}$, and for each $c_i, r_j \in X$ such that (1) $(c_i, r_j) \in E_A$ or (2) $(r_j, c_i) \in E_B$, we add r_{jc_i} and the binary constraint between r_{jc_i} and r_j and the unary constraint that player 2 cannot use r_{jc_i} for (1) and add c_{ir_j} and the binary constraint between c_{ir_j} and c_i and the unary constraint that player 1 cannot use c_{ir_j} for (2) to $\overline{X}$. The construction will result in a tree decomposition for the new RGG with polynomial increase in the treewidth.

We now need to show it is a proper tree decomposition. That is (a) $\bigcup_{X \in B} X = V$, (b) for each $e = \{e_1, e_2\} \in E$, $\exists X \in B$ such that $e_1, e_2 \in X$, and (c) for any X, X', and $X'' \in B$, if X' is on the path of X and X'', then $X \cap X'' \subseteq X'$.

(a) First, need to show that $\bigcup_{\overline{X} \in \overline{B}} \overline{X} = R \cup C \cup \overline{R} \cup \overline{C} \cup Const_1 \cup Const_2 \cup \overline{Const}_1 \cup \overline{Const}_2$ where $Const_1$ ($\overline{Const}_1$) and $Const_2$ ($\overline{Const}_2$) are the original (new) constraints of player 1 and player 2, respectively. Clearly, R, C, $Const_1$, and $Const_2$ are in the tree decomposition since they are in the tree decomposition of T. Note that the $Const_1$ and $Const_2$ remain the same in the new polytopes. For each $r_i \in R$, and $c_j \in Pa(r_i)$ appear together at least once in some X of B, by our construction r_{ic_j}, the unary and binary constraints appear in the new tree $\overline{B}$. (similarly for other) Thus, the new tree decomposition consists of all of the nodes of the new RGG.

(b) Second, we need to show that all of the edges in the new RGG is also in some X together. (1) The edges between the old resources and the old constraints are already in the new tree decomposition. (2) The edges between the old resources and the new resources are in the new tree decomposition from our construction by add new resource when we see a pair of old resource connection. (3) The edges between the old resources and the new constraints are in the new tree decomposition because add the constraints when we add the new resources (this also add in edges of the new resources of the new constraints).

(c) The original resource nodes and the constraint nodes already satisfy the intersection property. The new resource nodes and (unary and binary) constraint nodes only appear when two original resources appear together, which also satisfy the intersection property. □

Example 4. *Security games (Section 4.1) are special cases of bilinear games. Suppose now an attack on each target t will also spread to neighboring targets, with the attacker receiving the sum of utilities from targets in the neighborhood of t. The resource graph will be similar to Fig. 1(a), with additional edges to attacker node at target t from defender nodes from neighboring targets. The game is bilinear. If the neighbor relationship graph has constant treewidth, the constraint resource graph also has constant treewidth, and hence Theorem 2 applies.*

References

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

[2] S. Barman. Approximating nash equilibria and dense bipartite subgraphs via an approximate version of caratheodory's theorem. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 361–369, 2015.

[3] S. Barman, K. Ligett, and G. Piliouras. Approximating nash equilibria in tree polymatrix games. In *International Symposium on Algorithmic Game Theory*, pages 285–296. Springer, 2015.

[4] H. Chan and A. X. Jiang. Congestion games with polytopal strategy spaces. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 165–171. AAAI Press, 2016.

[5] H. Chan, A. X. Jiang, K. Leyton-Brown, and R. Mehta. Multilinear games. In *Web and Internet Economics: 12th International Conference, WINE 2016, Montreal, Canada, December 11-14, 2016, Proceedings*, pages 44–58, 2016.

[6] X. Chen and X. Deng. Settling the complexity of 2-player Nash-equilibrium. In *FOCS: Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, pages 261–272, 2006.

[7] C. Daskalakis, A. Fabrikant, and C. Papadimitriou. 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*, pages 513–524, 2006.

[8] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. In *STOC: Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 71–78, 2006.

[9] C. Daskalakis, G. Schoenebeck, G. Valiant, and P. Valiant. On the complexity of nash equilibria of action-graph games. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 710–719, 2009.

[10] A. Del Pia, M. Ferris, and C. Michini. Totally unimodular congestion games. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 577–588, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics.

[11] A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *STOC: Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 604–612. ACM New York, NY, USA, 2004.

[12] J. Garg, A. X. Jiang, and R. Mehta. Bilinear games: Polynomial time algorithms for rank based subclasses. In *Proceedings of the 7th International Conference on Internet and Network Economics*, WINE'11, pages 399–407, 2011.

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

[14] A. Jiang, H. Chan, and K. Leyton-Brown. Resource graph games: A compact representation for games with structured strategy spaces. In *AAAI Conference on Artificial Intelligence*, 2017.

[15] A. Jiang and K. Leyton-Brown. Computing pure Nash equilibria in symmetric action graph games. In *AAAI: Proceedings of the AAAI Conference on Artificial Intelligence*, pages 79–85, 2007.

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

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

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

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

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

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

[22] D. Korzhyk, V. Conitzer, and R. Parr. Security games with multiple attacker resources. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 273, 2011.

[23] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 2011.

[24] J. F. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

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

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

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

[28] R. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

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

[30] Y. Vorobeychik. *Mechanism Design and Analysis Using Simulation-Based Game Models*. PhD thesis, University of Michigan, 2008.