An FPTAS for Computing Nash Equilibrium in Resource Graph Games

Hau Chan¹ and Albert Xin Jiang²

¹ University of Nebraska-Lincoln, Lincoln, NE 68588
² Trinity University, San Antonio, TX 78212
hchan3@unl.edu, xjiang@trinity.edu

Abstract

We consider the problem of computing a mixed-strategy Nash equilibrium (MSNE) in resource graph games (RGGs), a compact representation for games with an exponential number of strategies. At a high level, an RGG consists of a graphical representation of utility functions together with a representation of strategy spaces as convex polytopes. RGGs are general enough to capture a wide variety of games studied in literature, including congestion games and security games.

In this paper, we provide the first Fully Polynomial Time Approximation Scheme (FPTAS) for computing an MSNE in any symmetric multilinear RGG where its constraint moralized resource graph has bounded treewidth. Our FPTAS can be generalized to compute optimal MSNE, and to games with a constant number of player types. As a consequence, our FPTAS provides new approximation results for security games, network congestion games, and bilinear games.

1 Introduction

There has been increasing interest in using game theory to model real-world multi-agent 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. In particular, in many domains [Daskalakis et al., 2006a; Rabinovich et al., 2009; Immorlica et al., 2011; Korzhyk et al., 2010], 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. Nevertheless, this space of pure strategies is often structured, meaning that it can be represented compactly.

Recently, Jiang et al., 2017] 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 studied in literature. In an RGG, there are n players and a set A of resources. A pure strategy of a player is a subset of the resources, represented by an |A|-dimensional 0-1 vector. Each player’s set of pure strategies is represented compactly as a polytopal strategy space: the set of integer points in a convex polytope, specified by a set of linear inequality constraints. There is a resource graph, a directed graph whose vertices are the resources A, and the graph depicts the utility structure of the game. RGGs are able to compactly encode a wide variety of games studied in literature, including security games [Korzhyk et al., 2010; Tambe, 2011], network congestion games [Daskalakis et al., 2006a], simultaneous auctions [Rabinovich et al., 2009], dueling algorithms [Immorlica et al., 2011], and Blotto games [Ahmadianjad et al., 2016].

In this paper, we consider the problem of computing a mixed-strategy Nash equilibrium (MSNE) in RGGs. RGGs can represent arbitrary games, as a result the PPAD-hardness of finding an MSNE in normal form games [Chen and Deng, 2006; Daskalakis et al., 2006b] implies the PPAD-hardness for finding MSNE in RGGs. 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 that there is a FPTAS for computing MSNE for a symmetric multilinear RGG whose constraint moralized resource graph has bounded treewidth. Treewidth is a well-studied graph property that has been applied to many graph-related problems, including Bayesian network inference and constraint satisfaction. In particular a tree has treewidth of 1. 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 results for these games.
In particular, (i) we provide FPTAS for computing MSNE in generalizations of security games [Korzhyk et al., 2011b; 2011a], in particular for cases when the defender and attacker have complex strategy sets not covered by existing results. (ii) [Chan and Jiang, 2016] provided a FPTAS for MSNE in a class of congestion games [Rosenthal, 1973] where the players’ strategy constraints are totally unimodular, and the constraint graph has bounded treewidth and bounded degree. Our FPTAS applied to congestion games only requires bounded treewidth, without restriction on degree, and as a result is applicable to a broader range of congestion games. (iii) Bilinear games [Garg et al., 2011] 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. Furthermore, due to the special structure of bilinear game utilities, we can avoid the moralization step, resulting in further reduced treewidth.

Our algorithm makes use of a compact, primal-dual formulation of the MSNE condition. We then use a tree-decomposition based dynamic programming approach to solve the resulting feasibility problem, passing partial assignments of primal and dual variables and partial sums of constraints across subgraphs. We omit all proofs in this version. A longer version of the paper with proofs and more discussion is available on the authors’ websites.

2 Preliminaries

Denote by $\mathbb{Z}_+$ the set of nonnegative integers. A rational polytope is defined by a set of inequalities with integer coefficients; $P = \{ x \in \mathbb{R}^n \mid D x \leq f \}$ 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 $u_i : S \rightarrow \mathbb{R}$. A mixed strategy $\sigma_i$ of player $i$ is a probability distribution over her pure strategies. Denote by $\sigma = (\sigma_1, \ldots, \sigma_n)$ a mixed strategy profile. Denote by $\sigma_{-i}$ the mixed strategy profile of players other than $i$. Denote by $u_i(\sigma)$ the expected utility of player $i$ under $\sigma$.

For $\epsilon \geq 0$, player $i$’s mixed strategy $\sigma_i$ is an $\epsilon$-best response to $\sigma_{-i}$ if $u_i(\sigma_i, \sigma_{-i}) \geq u_i(s_i', \sigma_{-i}) - \epsilon$ for all $s_i' \in S_i$. A mixed strategy profile $\sigma$ is an $\epsilon$-MSNE if for each player $i \in N$, $\sigma_i$ is an $\epsilon$-best response to $\sigma_{-i}$. 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 $\Gamma = (N, A, \{S_i\}_{i=1}^n, G, \{u^\alpha\}_{\alpha \in A})$, where $N = \{1, 2, \ldots, n\}$ is the set of $n$ players; $A = \{1, \ldots, m\}$ is the set of $m$ resources. Each $s_i \in S_i$ is an $|A|$-dimensional 0-1 vector. $S_i$ is represented as a polytopal strategy space where $S_i = P_i \cap \{0, 1\}^{|A|}$, $P_i = \{x \in [0, 1]^{|A|} \mid D_i x \leq f_i\}$, $D_i \in \mathbb{Z}_+^{|A|}$, and $f_i \in \mathbb{Z}_+$. In other words, $P_i$ is a rational polytope defined by $l_i$ linear constraints. We let $s_{i\alpha}$ denote the component corresponds to $\alpha \in A$. Given a pure-strategy profile $s = (s_1, \ldots, s_n)$, the configuration $c = \sum_{i \in N} s_i$ is an integer vector that counts the number of players who have selected each resource. The resource graph $G = (A, E)$ is a directed graph that can contain self-loops. The neighborhood of $\alpha \in A$, denoted by $N(\alpha)$, is the set of resources with edges going into $\alpha$. Let $C(\alpha) \subseteq \mathbb{Z}_+^{|A|}$ be the set of local configurations over $\alpha$, and $c(\alpha) \in C(\alpha)$ a local configuration. The local utility function $u^\alpha : C(\alpha) \rightarrow \mathbb{R}$ represents the utility contribution of using $\alpha$ given the configuration over its neighborhood $\nu(\alpha)$. Given the pure-strategy profile $s = (s_1, \ldots, s_n)$, the utility of player $i$ is defined to be $u_i(s) = \sum_{\alpha \in A} s_{i\alpha} u^\alpha(c(\alpha))$.

[Jiang et al., 2017] showed that the representation size is polynomial in $m$, $n^2$, and $\sum l_i$, where $I$ is the maximum in-degree of the resource graph.

We define symmetric RGGs to be those in which the players have the same polytopal strategy space.

Definition 1. An RGG, $\Gamma = (N, A, \{S_i\}_{i=1}^n, G, \{u^\alpha\}_{\alpha \in A})$, 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 $D \in \mathbb{Z}_+^{l_i \times |A|}$ and $f \in \mathbb{Z}_+^l$.

More generally, we call an RGG $k$-symmetric if there are $k$ equivalence classes of players, and within each equivalence class the players have the same polytopal strategy space.

Definition 2. An RGG, $\Gamma = (N, A, \{S_i\}_{i=1}^n, G, \{u^\alpha\}_{\alpha \in A})$, is $k$-symmetric if and only if the players are partitioned into $k$ classes, and for any player $i$ in class $j \in \{1, \ldots, k\}$, $D_i = D^j$, $f_i = f^j$ for some $D^j \in \mathbb{Z}_+^{l_i \times |A|}$ and $f^j \in \mathbb{Z}_+^l$ (and therefore $S_i = S^j$, and $P_i = P^j$).

Every ($k$-)symmetric game has a ($k$-)symmetric MSNE in which every player in the same equivalence class plays the same (possibly mixed) strategies [Nash, 1951].

2.3 Multilinearity and Marginal Vectors

When $|S_i|$ is exponential, representing a mixed strategy $\sigma_i$ explicitly would take exponential space. Given player $i$’s mixed strategy $\sigma_i$, the marginal vector $\pi_i$ that corresponds to $\sigma_i$ is $\pi_i = \tilde{E}_c | s_i = \sum_{s_j \in S_j} \sigma_j(s_j) s_j$. In other words, $\pi_{i\alpha}$ is the marginal probability that resource $\alpha$ is chosen under the mixed-strategy $\sigma_i$. 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. [Chan et al., 2016] showed that this can be done if the game has the multilinearity property: for all $i,j \in N$, given a fixed $s_{-j}$, $u_i$ is a linear function of $s_j$. Not all RGGs are multilinear; the following proposition gives a sufficient condition for an RGG to be multilinear.

Proposition 1 (Proposition 7 of [Jiang et al., 2017]). An RGG, $\Gamma = (N, A, \{S_i\}_{i=1}^n, G, \{u^\alpha\}_{\alpha \in A})$, is multilinear if, for each player $i$, for each $\alpha \in A$, 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.

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 manner that requires less space. [Chan et al., 2016] 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 $\pi_i$, we can efficiently generate a mixed strategy $\sigma_i$, represented as a sparse vector with polynomial number of nonzero entries, such that $\pi_i = \sum_{s_i \in S_i} \sigma_i(s_i)s_i$.

3 Statements of Results

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

Assumption 1. $\forall i \in N$, the matrix $D_i$ is totally unimodular.

Assumption 2. For each $i \in N$, for each $\alpha \in A$, and for each $s_i \in S_i$, $\sum_{\alpha' \in \alpha} s_i \alpha' \leq 1$.

Without either one of these assumptions, even single agent versions of RGG can represent NP-hard optimization problems, i.e., we end up with problems whose hardness has nothing to do with game theory.

Assumption 3. For $\alpha \in A$, the range of $u^\alpha$ is in $[0, 1]$. A fully-polynomial time approximation scheme (FPTAS) for MSNE is an algorithm that given an RGG, computes an $\epsilon$-MSNE in time $\text{poly}(m^2, n, \epsilon, l)$. While Nash equilibria are preserved under affine transformations of the utility functions $u_i$, Nash equilibria are not. E.g. multiplying all $u_i$ by a constant $\kappa$ would turn a $\kappa$-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 $u_{\max} = \max_{s_i} u_i(s_i)$ and $u_{\min} = \min_{s_i} u_i(s_i)$ are the maximum and minimum achievable utilities of the game, respectively. Then use the normalized game to measure the quality of approximation, i.e., $\epsilon$. An $\epsilon$-MSNE in the original game would correspond to an $(u_{\max} - u_{\min})/\epsilon$-MSNE in the normalized game. While Assumption 3 implies $u_i(s_i) \in [0, 1]$ 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 [Barman et al., 2015]. 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. $(u_{\max} - u_{\min}) \geq \frac{1}{\text{poly}(n, m, \sum_i |\alpha|)}$.

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 games discussed in this paper.

Constrained Moralized Resource Graph. Let $C = \{1, \ldots, l\}$ be the set of constraints, corresponding to rows of the constraint matrix $D$ of a symmetric RGG.

Figure 1: $T$-target Security Games. (a. Top Left) RGG representation of Security Games, (b. Top Right) Constraint Moralized Resource Graph, and (c. Bottom) a tree decomposition of the Constraint Moralized Resource Graph.

Definition 3. Given a resource graph $G = (A, E)$, its normalized resource graph is an undirected graph $MG = (A, \overline{E} \cup E')$ where $\overline{E}$ is an undirected edge set of $E$, and $E' = \{(\alpha, \alpha') \in A \times A \mid \exists \alpha \in A \ s.t. (\alpha, \alpha), (\alpha', \alpha) \in E\}$ is a set of undirected edges connecting the parents of their children. The constraint moralized resource graph is an undirected graph $CMG = (A \cup C, \overline{E} \cup E' \cup E'')$ where $E'' = \{(c, \alpha) \in C \times A \mid D_{ca} \neq 0\}$ 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.

Our main result is stated as follows.

Theorem 1. There is an FPTAS for computing a $k$-symmetric MSNE in $k$-symmetric RGGs that satisfy Assumptions (1-4) and have constraint moralized resource graphs with bounded treewidth, when $k$ is a constant. In particular, the algorithm finds an $\epsilon$-MSNE in time $\text{poly}(n, m, l, \frac{1}{\epsilon})$. The algorithm can be extended to compute $\epsilon$-MSNE with the highest (or lowest) social welfare.

Applications to Security Games. In a typical security game [Tambe, 2011] 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. Defender and attacker’s utilities depend on whether the attacked target is covered. [Jiang et al., 2017] showed that security games can be compactly encoded as RGGs. Figure 1(a) illustrates the resource graph. For each target $t \in T$, we have a resource node $a_t$ for the attacker, representing the utility of attacker for attacking target $t$, and two resource nodes $b_{ot}$ and $b_{lt}$ for the defender, representing the defender’s utility for not protecting and protecting $t$, respectively. The constraint matrices of the attacker and the defender are totally unimodular, and the strategies of the players satisfy Proposition 1. This can be generalized to the model of [Korzhyk et al., 2011a] where the attacker can attack $K$ targets.

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 provide a constant-width tree decomposition of the moralized constraint resource graph in Figure 1(c). Since we have only two classes of players (an attacker and a defender) and constant treewidth, our algorithm yields an FPTAS for Nash equilibrium in these games.
Our result holds for general classes of security games with arbitrary strategy constraints for the players as long as their constraint matrices are totally unimodular and the treewidth of the moralized constraint resource graph is bounded.

**Proposition 2.** 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 2 applies.

**Applications to 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.

Each resource in the congestion game corresponds to a resource node in the RGG representation. The resource graph contains only self-loops. When the constraint matrices for both players are totally unimodular, and the treewidth of the constraint resource graph (without moralization) is bounded, there is an FPTAS for computing MSNE.

**Example 2 (Network Congestion).** Consider network congestion games [Fabrikant et al., 2004], where resources correspond to edges in a network, and each player chooses a path from a source to a destination in the network. [Chan and Jiang, 2016] 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 \( a_i \) 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.

**Bilinear Games.** A bilinear game [Garg et al., 2011] is played between two players 1 and 2. The strategy spaces of the players are described using polytopes \( X \) and \( Y \). Player 1 and player 2 have payoff matrices \( A \in \mathbb{R}^{M \times N} \) and \( B \in \mathbb{R}^{M \times N} \), respectively. In particular, given (pure) strategies \( x \in X \) and \( y \in Y \), \( u_1(x, y) = x^T Ay \) and \( u_2(y, x) = x^T By \).

Given a bilinear game with (possibly sparse) \( A \) and \( B \), the resource graph is a bipartite graph where one side has \( M \) resources corresponding to rows and the other side has \( N \) resources corresponding to columns. There is an edge from column \( c \) to row \( r \) iff \( A_{r,c} \neq 0 \), and an edge from row \( r \) to column \( c \) iff \( B_{r,c} \neq 0 \). We can further exploit the linearity of the utility functions, and as a result we do not need to moralize the resource graph.

**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.

### 4 Sketch of Derivation of Our FPTAS

#### 4.1 A Primal-Dual Definition of MSNE

We can express the expected utility as \( \pi_i(\pi) = \pi_i^T \nabla_i(\pi) \) where \( \nabla_i(\pi) \) is the utility gradient of \( i \) and \( \nabla_i(\pi)\alpha \) denotes the expected utility contribution of \( i \) from using the resource \( \alpha \) and \( \pi_i^T \) is the transpose of \( \pi_i \). [Jiang et al., 2017] showed that given an RGG and \( \pi \), the utility gradient \( \nabla_i(\pi) \) can be computed in polynomial time. Moreover, to compute \( \nabla_i(\pi)\alpha \) we only need to know \( \langle \pi\alpha' \rangle_{j \neq i, \alpha' \in \nu(\alpha)} \), i.e., other players’ marginals projected to \( \alpha \)'s neighborhood.

In the standard definition of MSNE, we need to enumerate all pure strategies to verify whether a mixed-strategy profile is a MSNE. This is not feasible for games (RGGs) with exponential number of pure strategies. We need a more efficient way to check whether a mixed-strategy profile is an MSNE.

Given the marginal vectors of others \( \pi_{-i} \), the best response for \( i \) is the solution to the following primal and dual LPs.

**Primal**

\[
\begin{align*}
\text{maximize} & \quad \pi_i^T \nabla_i(\pi_{-i}) \\
\text{subject to} & \quad D_i \pi_i \leq f_i \\
& \quad \pi_{\alpha} \geq 0 \quad \text{for } \alpha \in \mathcal{A} \\
& \quad \lambda_{ij} \geq 0 \quad \text{for } j = 1, 2, \ldots, l_i
\end{align*}
\]

**Dual**

\[
\begin{align*}
\text{minimize} & \quad f_i^T \lambda_i \\
\text{subject to} & \quad D_i \pi_i \leq f_i \\
& \quad \lambda_{ij} \geq 0 \quad \text{for } j = 1, 2, \ldots, l_i
\end{align*}
\]

Here we assume that the constraint \( \pi_{\alpha} \leq 1 \) is embedded into the constraints \( D_i \pi_i \leq f_i \). Given that the primal LP has a solution, the primal and dual LPs solutions have the same optimal objective value. As a result, \( \pi_i \) is its best response when there exists a feasible dual vector \( \lambda_i \) such that \( \pi_i^T \nabla_i(\pi_{-i}) = f_i^T \lambda_i \). Thus, a feasible marginal vector \( \pi \) is an MSNE if for each player \( i \), there exists \( \lambda_i \geq 0 \) such that...
\[ D^T_i \lambda_i \geq \nabla_i(\pi_i) \quad \text{and} \quad \pi^T_i \nabla_i(\pi_i) = f^T_i \lambda_i. \] 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 \( \pi_i. \)

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

**Lemma 1.** For any \( \delta > 0 \), there is a (symmetric) marginal vector \( \forall q_i = (q_{i\alpha})_{\alpha \in A} \), in which \( q_{i\alpha} \in \{0, \frac{1}{2}, \frac{3}{2}, \ldots, 1\} \) and \( K = O\left(\frac{\log m}{\delta}\right) \), such that \( u(q_i, q_{i-1}) = q^T_i \nabla(q_{i-1}) \geq q^T_i \nabla(q_{i-1}) - O(\delta dm) \) for \( q_i \in [0, 1]^m \) that satisfies \( Dq^T \leq f \) and \( d \) is the maximum in-degree of the resource graph \( G. \)

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

### 4.2 Efficient Algorithm to Compute \( \epsilon \)-MSNE

From Lemma 1, there is a marginal vector \( q_i \) that lies in the \( m \)-dimensional uniform grid with discretization size \( \frac{1}{O(\delta dm)} \). Thus, a brute-force approach does not yield an efficient algorithm. We instead use the definition of \( \epsilon \) in terms of linear programming of Section 4.1 and introduce a message-passing algorithm that runs on a tree decomposition of the constraint relaxed resource graph.

**Discretization of Variables.** To search for the marginal representation, we provide discretization bounds for the existence of an \( \epsilon \)-MSNE in the marginal and utility spaces. Here, we provide other discretization spaces that are necessary for our algorithm. We use the notation \( D_T \) to denote the subcomponent of \( D \) consisting of the rows in \( C \subseteq C \) and columns in \( A \subseteq A \). If \( |C| = 1 \) and \( |A| = 1 \), then \( D_T \) is the value corresponding to the row and column. To begin, recall that we want to find a marginal vector \( q_i = (q_{i\alpha})_{\alpha \in A} \) where \( q_{i\alpha} \in \delta \) (as in Lemma 1) such that \( Dq_i \leq f \), \( D_T^T \lambda_i \geq \text{proj}(\nabla(q_{i-1})) \), and \( q^T \text{proj}(\nabla(q_{i-1})) \geq f^T \lambda_i - \frac{\epsilon}{2} - O(\delta dm) \) for \( \lambda = (\lambda_{\alpha})_{\alpha \in C} \) where \( \lambda_{\alpha} \in \Lambda \).

Clearly, we are not going to try all possible \( q_i \) in \( Q_i \) directly. Instead, our approach is to find feasible \( (q_{i\alpha})_{\alpha \in A} \) and \( (\lambda_{\alpha})_{\alpha \in C} \) for some ordering of \( A \subseteq A \) and \( C \subseteq C \) induced by the tree decomposition incrementally. Thus, we need to keep track of the partial sums of (1) \( D_T q_i \) for each constraint \( c \), (2) \( D_T^T \lambda_i \) for each resource \( \alpha \), (3) \( q^T \text{proj}(\nabla(q_{i-1})) \), and (4) \( f^T \lambda_i \) for a given \( q \) and \( \lambda_i \). Because \( D \) is totally unimodular, its entries can only have values of 0, 1, or -1. Thus, the possible values of the partial sums are finite. For (1), given a constraint \( c \), we let \( \delta S_c \) be the partial sum of the constraint from \( S_c \). For (2), given a resource node \( \alpha \), we let \( \delta \lambda_{\alpha} \) be the partial sum of a resource constraint \( \alpha \) (i.e., \( D_T^T \lambda_i \) from \( S_{\alpha} \)). For (3), we let \( \delta f \) to be the partial dual objective sum from \( F \). Finally, to ensure we have a marginal vector of \( \epsilon \)-Nash equilibrium, we let \( \delta = O\left(\frac{\epsilon}{\delta dm}\right) \). For the sake of space, we leave the details of the discretization of set in full version.

**Message Passing Algorithm.** Given the tree decomposition of the constraint relaxed marginal graph and the discretization spaces, we provide a two-pass 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 \( A_X \) and \( C_X \), respectively. From the moralization and the tree decomposition, for each \( \alpha \in A \), 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 \( q_i \cdot \text{proj}(\nabla(\nu(\alpha))) \). Because of that, we precompute the tree and designate a node of \( X \) to compute the partial sum of the expected utility for each resource \( \alpha \) exactly once. Of course, the designated node \( X \) should contain the \( \nu(\alpha) \). We let \( A_X \subseteq A_X \) to denote the set of resources where we compute the expected utility at \( X \in B \). Such preprocessing can be done using depth-first search or breath-first search.

We first present the algorithm when the tree \( T \) is a line. Then, we show how we can generalize the algorithm to a tree. We assume all of the graphs are connected, otherwise we can just form a new tree by joining the connected components.

**The Line Case.** Suppose that the resulted tree decomposition is a line, say \( L = (B, E) \). We relabel the nodes of \( L \) so that \( B = \{1, 2, \ldots, n\} \) where 1 is the leftmost node and \( n \) is the right most node. Thus, the edge set \( E = \{i, i + 1, i = 1, \ldots, n - 1\} \) 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 \( \epsilon \)-MSNE, and the goal of downstream pass is to construct a feasible \( \epsilon \)-MSNE.

**Upstream pass.** For each node \( i \in B \), we construct a set \( f_i \) containing tuples of \( Q_i \times C_{i-1} \times S^C_{i-1} \times S^A_{i-1} \times T \times F \) based on \( M_{i-1} \), which is the message \( i-1 \) sends to \( i \). More specifically,

\[ f_i = \{(q_{i\alpha})_{\alpha \in A_i}, \partial S_{i-1}, \partial C_{i-1}, \partial \lambda_{i-1} \}_{\alpha \in A_i}, \theta, \delta \}

\[ \in Q_i \times C_{i-1} \times S^C_{i-1} \times S^A_{i-1} \times T \times F \]

\[ \theta = \sum_{\alpha \in A_i} q_{i\alpha} \text{proj}(\nabla(q_{i\alpha})) + \frac{\epsilon}{2} \tag{1} \]

\[ \forall c \in C \cap C_{i+1}, \partial S_c = D_{c, A_i} \cap C_{i+1} \cap A_i + 1_{\{n\in \text{set} \mid \text{set} \}} \tag{2} \]

\[ \forall c \in C \cap C_{i+1}, \partial S_c = D_{c, A_i} \cap C_{i+1} + 1_{\{n\in \text{set} \mid \text{set} \}} \tag{3} \]

\[ \forall c \in C \cap C_{i+1}, \partial S_c = D_{c, A_i} \cap C_{i+1} \cap A_i + 1_{\{n\in \text{set} \mid \text{set} \}} \tag{4} \]

\[ \forall c \in C \cap C_{i+1}, \partial S_c = D_{c, A_i} \cap C_{i+1} \cap A_i + 1_{\{n\in \text{set} \mid \text{set} \}} \tag{5} \]

\[ \forall c \in C \cap C_{i+1}, \partial S_c = D_{c, A_i} \cap C_{i+1} \cap A_i + 1_{\{n\in \text{set} \mid \text{set} \}} \tag{6} \]

\[ \forall c \in C \cap C_{i+1}, \partial S_c = D_{c, A_i} \cap C_{i+1} \cap A_i + 1_{\{n\in \text{set} \mid \text{set} \}} \tag{7} \]

\[ \forall c \in C \cap C_{i+1}, \partial S_c = D_{c, A_i} \cap C_{i+1} \cap A_i + 1_{\{n\in \text{set} \mid \text{set} \}} \tag{8} \]

\[ \forall c \in C \cap C_{i+1}, \partial S_c = D_{c, A_i} \cap C_{i+1} \cap A_i + 1_{\{n\in \text{set} \mid \text{set} \}} \tag{9} \]
where \& denotes the regular “and” condition and \(1_{\text{exp}}\) is the indicator function for the expression (1 if the exp is true, 0 otherwise).

The (1) condition keeps track of the partial sums of the expected utilities of the resources in \(A_i^j\), \(j = 1\) to \(i-1\), and add the expected utilities of resources in \(A_i^j\). 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 \(D_{C_i,A_i} = g_{A_i} \leq f_{C_i}\)). 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 \(\partial S_c\) for \(c\) that appears in the previous \(j = i-1, \ldots, \text{nodes}\).

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., \(D^*_A,c_i,\lambda_{C_i} \geq \text{proj}(\nabla(q_{A_i}))_{A_i}\)). Clearly, we can only compute the projected expected utility for the resources in \(A_i^j\). 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 \(\alpha\) in \(A_i^j\). The reason we are doing this is because we will not be 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 \(\alpha\).

For condition (8), we keep track of the partial sum of \(f_{\alpha,C_i,\lambda_{C_i}}\). 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 \(\text{poly}(n, m, l, 1/\epsilon)\), if the width of the tree decomposition is bounded by a constant.

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

\[
M_{i\rightarrow i+1} = \{((q_{A_i})_{\alpha_a \in A_i}, (\partial S_{C_i})_{c \in C_i}, (\lambda_{C_i})_{c \in C_i}, (\partial \alpha)_{\alpha_\alpha \in A_i},
\partial t, \partial f) \in f_i \mid \forall c \in C_i \setminus C_{i+1}, \partial S_{C_i} \leq f_c \} & & \text{if } i > 0,
\forall \alpha_a \in A_i \setminus C_{i+1}, \partial \alpha_a \geq 0 \} \cup \{f_i\}
\Rightarrow \partial t \geq \partial f - \epsilon
\]

\(M_{i\rightarrow i+1}\) and the message from \(i\) to \(i+1\) is

\[
M_{i\rightarrow i+1} = \{(q_{A_i})_{\alpha_a \in A_i}, (\partial S_{C_i})_{c \in C_i}, (\partial \alpha)_{\alpha_\alpha \in A_i},
(\lambda_{C_i})_{c \in C_i}, (\partial t, \partial f) \} \forall (q_{A_i})_{\alpha_a \in A_i}, (\partial S_{C_i})_{c \in C_i}, (\partial \alpha)_{\alpha_\alpha \in A_i}, (\partial t, \partial f) \in f_i.
\]

When \(i = 0\), \(C_0 = A_0 = C_{n+1} = A_{n+1} = \emptyset\) and \(M_{0\rightarrow 1} = \{0, 0, 0, 0, 0\}\). When \(i = n\), \(C_{n+1} = A_{n+1} = \emptyset\) and \(n\) does not send message to \(n+1\) but still construct table \(M_{fn}\).

**Downstream Pass.** Now that we have completed the upstream pass, we have enough information to construct an \(\epsilon\)-MSNE. We will do this by starting selecting partial \(\epsilon\)-MSNE for \(n, n-1, \ldots, 1\), sequentially. In particular, given the selected partial \(\epsilon\)-MSNE at node \(i, i\) will send node \(i-1\), \(R_{i\rightarrow i-1}\), the feasible strategies that \(i-1\) should select to ensure \(\epsilon\)-MSNE. The details are in full version\(^1\).

**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 \(f_i, M_{i\rightarrow \text{parent}}, R_i, R_{i\rightarrow \text{left}}, R_{i\rightarrow \text{right}}\) 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 \(M_{i\rightarrow \text{parent}}\) and \(M_{i\rightarrow \text{parent}}\) 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 \(R_{i\rightarrow \text{left}}\) and \(R_{i\rightarrow \text{right}}\) can be constructed the same way but based on the messages \(M_{i\rightarrow \text{left}}\) and \(M_{i\rightarrow \text{right}}\) that satisfy the conditions (1)-(8).

**Computing Optimal MSNE.** For example, to find the social welfare maximizing symmetric \(\epsilon\)-MSNE in the downstream pass, at the root we select a tuple with the highest \(\partial t\), i.e., utility, among the feasible tuples. Continuing the downstream pass as before yields a symmetric \(\epsilon\)-MSNE with maximum social welfare.

**RGGs with Constant Number of Player Types.** We can generalize our message passing algorithms to cases where RGGs have a constant number of types. It can be verified our results 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 \((q_{A_i})_{\alpha \in A_i}, (\partial S_{C_i})_{c \in C_i}, (\partial \alpha)_{\alpha \in A_i}, (\partial t, \partial f)\) for each class \(j = 1, \ldots, k\). This results in running time and table size exponential in \(k\) and the treewidth.

**References**


