On the Approximation of Nash Equilibria in Sparse Win-Lose Multi-player Games

Zhengyang Liu, [1] **Jiawei Li,**[2] **Xiaotie Deng** [2]
[1] Beijing Institute of Technology
[2] CFCS, Peking University
zhengyang@bit.edu.cn, davidlee1999@pku.edu.cn, xiaotie@pku.edu.cn

Abstract

A *polymatrix game* is a multi-player game over n players, where each player chooses a pure strategy from a list of its own pure strategies. The utility of each player is a sum of payoffs it gains from the two player's game from all its neighbors, under its chosen strategy and that of its neighbor. As a natural extension to two-player games (a.k.a. bimatrix games), polymatrix games are widely used for multi-agent games in real world scenarios.

In this paper we show that the problem of approximating a Nash equilibrium in a polymatrix game within the polynomial precision is PPAD-hard, even in sparse and win-lose ones. This result further challenges the predictability of Nash equilibria as a solution concept in the multi-agent setting. We also propose a simple and efficient algorithm, when the game is further restricted. Together, we establish a new *dichotomy* theorem for this class of games. It is also of independent interest for exploring the computational and structural properties in Nash equilibria.

Introduction

Nash equilibrium models and predicts the strategic behavior of selfish yet rational agents. Essentially, at any Nash equilibrium, the strategy of each agent is the best response with respect to the strategies from other agents. The celebrated Nash Theorem (Nash 1951, 1950) establishes the existence of such an equilibrium solution in every finite game, by leveraging fixed-point theorems. Due to its simplicity and elegance, Nash equilibrium has been widely applied to economics and political sciences, and to Internet studies in the fast growth human activities over it.

However, most of the Nash equilibrium results studied under variant settings are non-constructive, which means we cannot find such a stable state efficiently. Despite much effort made to develop algorithms to tackle this fundamental problem (Kuhn 1961; Lemke and J. T. Howson 1964; Janovskaya 1968; Wilson 1971; Howson Jr 1972; Garcia, Lemke, and Luethi 1973; Shapley 1974), the most widely used algorithm till now still needs an exponential time in the worst case (Savani and von Stengel 2006).

In his seminal work to characterize the proof structure of equilibrium problems (Papadimitriou 1994), Papadimitriou introduced the complexity class of PPAD, and proved that many equilibrium-related problems are contained in this class. The next major breakthrough (Daskalakis, Goldberg, and Papadimitriou 2009) developed Nash equilibrium in graphical games, simulation of arithmetic, logic operations and Brouwer's fixed points. This reduction allowed for a proof of finding an exponentially close approximate Nash equilibrium in a game over four or more players being PPAD-complete. Ultimately in the two player case, finding a polynomially small approximate solution is shown to be PPAD-complete by Chen et al. (Chen and Deng 2006; Chen, Deng, and Teng 2009).

This work studies the polynomial precision approximability issue for the problem of approximating a Nash equilibrium in a polymatrix game, focusing on the sparse and win-lose games. We restrict our discussion to games with a *succinct* representations. In particular, we focus on polymatrix games with n players where each player has two strategies. Each pair (i, j) of players play a small (2×2) bimatrix sub-game. The payoff of a player is the sum of the payoffs she gains in each sub-game, by its given strategy (the same strategy against all its opponents). We use a $2n \times 2n$ rational matrix $\mathbf{P}$ for a polymatrix game. For the PPAD-hardness result we obtain here, the restriction of the two by two pairwise subgame between every pair of players can easily be extended to more general m by m bimatrix subgame case for the same complexity result.

The polymatrix game model is much more important and interesting for two key points in equilibrium computation.

1. From the viewpoint of economics, bimatrix game is not the best suitable model in today's complicated environment, such as over the Internet connected society. The user profiling approach can allow us to model a wholistic agent action space. Such a real life observation motivates the adoption of the polymatrix game model in our study.

2. For the computational complexity issue, polymatrix games serves as the base from which further PPAD-hardness results are derived, such as other succinct games (Chen, Durfee, and Orfanou 2015), market equilibrium (Chen, Paparas, and Yannakakis 2017) and competitive equilibrium (Babichenko, Papadimitriou, and Rubinstein 2016).

Our Contributions We study a simple class of polymatrix games, called *Sparse Win-Lose polymatrix* (SWLP) games. Informally, given two positive integers a and b, an (a, b)-SWLP game is a polymatrix game if the payoff matrix satisfies the followings: Each row contains at most a ones, and each column contains at most b ones; the other entries are all zeros. In this paper, we establish a dichotomy theorem for Nash equilibrium computation of SWLP games. For the intractable aspect, we show that the problem of finding a Nash equilibrium in $(2, 2)$-SWLP games with polynomial (in terms of the number of players) precision is PPAD-hard. This surprising result strongly challenges the predictability of Nash equilibrium in the multi-agent setting due to the computational hardness in this extremely simple setting. For the positive part, we provide an efficient algorithm to compute an *exact* Nash equilibrium for other cases, i.e., $(1, n)$ or $(n, 1)$-SWLP games.

Related Work

Polymatrix games were introduced by (Janovskaya 1968) and its approximate computational complexity, within an exponentially small factor, was shown PPAD-hard implicitly, in the result of Daskalakis et al (Daskalakis, Goldberg, and Papadimitriou 2009). The approximation constant was relaxed further to be polynomially small (Chen, Deng, and Teng 2009). The followup work (Chen, Paparas, and Yannakakis 2017; Rubinstein 2015) improved the result by showing that the circuit problem is still hard even the precision is further relaxed. Chen et al. (Chen, Paparas, and Yannakakis 2017) proved that finding a $1/n$-well-supported Nash equilibrium in a polymatrix game is hard. It is worth mentioning that Rubinstein (Rubinstein 2015) showed that even if the polymatrix game is sparse and the precision parameter is an absolute constant, the result still holds. It is quite different from the bimatrix game setting (Lipton, Markakis, and Mehta 2003), unless PPAD can be solved within quasi-polynomial time. Very recently, Deligkas et al. (Deligkas, Fearnley, and Savani 2020) proved that finding a Nash equilibrium in tree polymatrix games with twenty actions per player is PPAD-hard.

However, the above complexity work didn't care about the simplicity of the payoff matrix. As we know, for two-player games, the hardness result depends on not only the sparsity (Chen, Deng, and Teng 2006) but also the simplicity of the payoff entries (Chen, Teng, and Valiant 2007)(Bilò and Mavronicolas 2019). Liu and Sheng (Liu and Sheng 2018) proved similar results to two-player games, they introduced chasing game as their base game, making the previous framework go through. For two-player games, base games like matching pennies game (Chen and Deng 2006; Chen, Deng, and Teng 2009; Daskalakis, Goldberg, and Papadimitriou 2009) and chasing game (Liu and Sheng 2018), enforce the sum of probabilities of every two successive strategies to be approximately equal. It is the crucial property to encode variables in the reductions. Considering the structure of a polymatrix game, it is natural that each player plays the same probability (that is exactly 1) on its own strategy set. To the end, we can avoid the logarithmic barrier in the polymatrix case. Note that the equilibrium computation of sparse win-lose bimatrix games still remains open.

From the positive result side, Deligkas et al. (Deligkas et al. 2017) showed that $(0.5 + \delta)$-approximate Nash equilibrium could be computed in time polynomial in the input size and $1/\delta$, with gradient descent method. Deligkas et al. (Deligkas, Fearnley, and Savani 2017) provided a deterministic QPTAS for polymatrix games with bounded treewidth and with logarithmically many actions per player. Barman et al. (Barman, Ligett, and Piliouras 2015) developed a quasi-polynomial time algorithm for approximating a Nash equilibrium in polymatrix games over trees. Ortiz and Irfan (Ortiz and Irfan 2017) provided an FPTAS for computing an approximate mixed-strategy Nash equilibrium in graphical multi-hypermatrix games. Efficient algorithms for win-lose games have also been studied in several settings, e.g., see (Chen, Deng, and Teng 2006)(Codenotti, Leoncini, and Resta 2006)(Datta and Krishnamurthy 2011)(Addario-Berry, Olver, and Vetta 2007).

Preliminaries

Notation. Throughout this paper, we use $[n]$ to denote the integer set $\{1, \dots, n\}$, and denote by $x = a \pm \epsilon$ the constraint that $x \in [a - \epsilon, a + \epsilon]$. All the vectors and matrices are in bold-face letters. We also use $\mathbf{0}$ to represent all zeros matrix of appropriate dimension. Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, we denote by $\mathbf{A}_i$ the i-th column of $\mathbf{A}$; given a vector $\mathbf{x} \in \mathbb{R}^n$, we use x_i to denote the i-th coordinate of $\mathbf{x}$.

An n-player two-action polymatrix game can be described by a $2n \times 2n$ payoff matrix $\mathbf{P}$, which is composed of $n \times n$ block matrices where each block is of size 2×2. The (i, j)-th block matrix represents the payoff for player j when playing the subgame with player i. The payoff for any player i is the sum of the payoff in n sub-games with all n players. As usual, we assume that the (i, i)-th matrix is zero matrix for any $i \in [n]$.

A *strategy profile* is represented by a rational-numbered vector $\mathbf{x} = (x_1, \dots, x_{2n}) \in \mathbb{R}_+^{2n}$, where x_{2i-1}, x_{2i} are the probabilities for player i to choose its first, second action respectively. We always have $x_{2i-1} + x_{2i} = 1$, for any $i \in [n]$. The expected payoff for player i choosing its first and second action are

$$u(2i-1, \mathbf{x}) := \mathbf{x}^T \cdot \mathbf{P}_{2i-1},$$
$$u(2i, \mathbf{x}) := \mathbf{x}^T \cdot \mathbf{P}_{2i},$$

where the $\mathbf{P}_t$ is the t-th column of $\mathbf{P}$.

For the rest of this paper, we use polymatrix games to denote two actions polymatrix games for brevity.

Instead of exact equilibrium, we introduce the well-supported Nash equilibrium. This solution concept allows us to compare a pair of strategies, approximately.

Definition 1 (ϵ-well-supported Nash Equilibrium)**.** A strategy profile $\mathbf{x}$ is an ϵ-well-supported Nash equilibrium for the polymatrix game $(n, \mathbf{P})$ if and only if for any $i \in [n]$:

$$x_{2i-1} = 0, \text{ when } u(2i-1, \mathbf{x}) < u(2i, \mathbf{x}) - \epsilon;$$
$$x_{2i} = 0, \text{ when } u(2i, \mathbf{x}) < u(2i-1, \mathbf{x}) - \epsilon.$$

We study the equilibrium computation of a very simple class of polymatrix games, named *sparse win-lose* polymatrix (SWLP) games.

Definition 2 ((a, b)-Sparse Win-Lose Polymatrix Game)**.** An (a, b)-sparse win-lose polymatrix game is a polymatrix game $(n, \mathbf{P})$ with following conditions:

1. The payoff matrix $\mathbf{P}$ has at most a non-zero entries in each row and at most b non-zero entries in each column.
2. Each entry in $\mathbf{P}$ is zero or one.

The starting point in our reduction is a well-known problem, named GCIRCUIT, which is proved to be PPAD-hard.

Definition 3 (GCIRCUIT (Chen, Deng, and Teng 2009))**.** The input of GCIRCUIT is a pair $(V, \mathcal{T})$, in which V is a set of nodes and $\mathcal{T}$ is a set of gates. Each gate $T \in \mathcal{T}$ is a 5-tuple $T = (G, v_1, v_2, v, \alpha)$, in which

1. $G \in \{G_+, G_-, G_=, G_<, G_\neg, G_\zeta, G_{\times\zeta}, G_\vee, G_\wedge\}$ is the type of the gate.
2. Nodes $v_1, v_2 \in V \cup \{\mathrm{nil}\}$ are the input of the gate, and $v_1 \neq v_2$ unless they are both nil.
3. The node $v \in V$ is the output of the gate, and $v \notin \{v_1, v_2\}$. We assume that each node can be used as output at most once.
4. $\alpha \in [0, 1]$ is a constant that will be used in gates $G_\zeta, G_{\times\zeta}$.

Each node $v \in V$ also corresponds to a real value variable $\mathbf{x}[v] \in [0, 1]$. We also called $\mathbf{x}[v]$ as the *value* of node v.[1] Each type of gate defines a constraint with respect to the value of input nodes, output node and approximate parameter ϵ. All the constraints are summarized in Table 1. For some cases that is not specified in Table 1, e.g. $|\mathbf{x}[v_1] - \mathbf{x}[v_2]| < \epsilon$ for the $G_<$ gate, the value of the output node could be arbitrary between 0 and 1.

The output of this problem is an assignment of $\mathbf{x}$ that satisfies all the ϵ-approximate gate constraints.

In this paper, we reduce an instance of GCIRCUIT to a $(2, 2)$-SWLP game, such that given any ϵ-well-supported Nash equilibrium, we can convert it to a valid output of the GCIRCUIT instance. We know that GCIRCUIT is PPAD-complete.

Lemma 1. *(Chen, Deng, and Teng 2009) Given a* GCIRCUIT $(V, \mathcal{T})$*, and* $\epsilon = \frac{1}{K^3}$*, where* $K = |V|$*, calculating an* ϵ*-approximate solution for* $(V, \mathcal{T})$ *is PPAD-complete.*

Review of the Former Proof

To make this paper self-contained, we briefly review the reduction from the GCIRCUIT problem to the polymatrix game in (Chen, Paparas, and Yannakakis 2017).

Let $(V, \mathcal{T})$ be a GCIRCUIT instance, where $|V| = K$. Let C be a one-to-one mapping from V to $[K]$, i.e. $C(v)$ is an index of vertex v. Let $n = 2K$ denote the number of players. For each gate $T \in \mathcal{T}$, we define two gadget matrices $\mathbf{L}[T], \mathbf{R}[T] \in [0, 1]^{n\times n}$, which simulate the functionality of the gate T. Then we construct the payoff matrix $\mathbf{P}$ as follows:

$$\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{A} & \mathbf{0} \end{pmatrix},$$

[1]We abuse the notation $\mathbf{x}$ for both the value of a node in GCIRCUIT and the strategy profile of all players in polymatrix game. When referring to the value of the node v, we will use square bracket to index it, like "$\mathbf{x}[v]$", to highlight the difference.

Given $T = (G, v_1, v_2, v, \alpha) \in \mathcal{T}$, $\epsilon \geq 0$; $\mathbf{x}[v] \in [0, 1]$ for any $v \in V$	
G_+	$\mathbf{x}[v] = \min(\mathbf{x}[v_1] + \mathbf{x}[v_2], 1) \pm \epsilon$
G_-	$\mathbf{x}[v] = \max(\mathbf{x}[v_1] - \mathbf{x}[v_2], 0) \pm \epsilon$
$G_=$	$\mathbf{x}[v] = \mathbf{x}[v_1] \pm \epsilon$
G_ζ	$\mathbf{x}[v] = \alpha \pm \epsilon$
$G_{\times\zeta}$	$\mathbf{x}[v] = \alpha\mathbf{x}[v_1] \pm \epsilon$
$G_<$	$\begin{cases} \mathbf{x}[v] = 1 \pm \epsilon & \text{if } \mathbf{x}[v_1] < \mathbf{x}[v_2] - \epsilon \\ \mathbf{x}[v] = 0 \pm \epsilon & \text{if } \mathbf{x}[v_1] > \mathbf{x}[v_2] + \epsilon \end{cases}$
$G_\neg$	$\begin{cases} \mathbf{x}[v] = 1 \pm \epsilon & \text{if } \mathbf{x}[v_1] \leq \epsilon \\ \mathbf{x}[v] = 0 \pm \epsilon & \text{if } \mathbf{x}[v_1] \geq 1 - \epsilon \end{cases}$
$G_\vee$	$\begin{cases} \mathbf{x}[v] = 1 \pm \epsilon & \text{if } \mathbf{x}[v_1] \geq 1 - \epsilon \text{ or } \mathbf{x}[v_2] \geq 1 - \epsilon \\ \mathbf{x}[v] = 0 \pm \epsilon & \text{if } \mathbf{x}[v_1], \mathbf{x}[v_2] \leq \epsilon \end{cases}$
$G_\wedge$	$\begin{cases} \mathbf{x}[v] = 1 \pm \epsilon & \text{if } \mathbf{x}[v_1], \mathbf{x}[v_2] \geq 1 - \epsilon \\ \mathbf{x}[v] = 0 \pm \epsilon & \text{if } \mathbf{x}[v_1] \leq \epsilon \text{ or } \mathbf{x}[v_2] \leq \epsilon \end{cases}$

Table 1: Constraints for gates in GCIRCUIT

where $\mathbf{A}, \mathbf{B} \in [0, 1]^{n\times n}$, $\mathbf{A} = \sum_{T\in\mathcal{T}} \mathbf{L}[T]$ and $\mathbf{B} = \sum_{T\in\mathcal{T}} \mathbf{R}[T]$. One can show that the final matrix $\mathbf{P}$ can satisfy all the constraints of gates in $\mathcal{T}$. We omit the original constructions of $\mathbf{L}[T], \mathbf{R}[T]$ here, and defer our new ones when used instead.

Let $(\mathbf{x}^*, \mathbf{y}^*) \in \mathbb{R}^{4K}$ be an ϵ-well-supported Nash equilibrium of polymatrix game $\mathbf{P}$, where $\mathbf{x}^*, \mathbf{y}^* \in \mathbb{R}^{2K}$ represent the strategy of the first and the last K players respectively. Then we take the value of node $\mathbf{x}[v] = {x^*}_{2C(v)-1}$ for all node $v \in V$. One can show that $\mathbf{x}[v]$ defined by the formula above is indeed an ϵ-approximate solution of GCIRCUIT instance $(V, \mathcal{T})$.

PPAD-hardness

In this section, we will prove our main theorem.

Theorem 2. *Finding any* $\frac{1}{n^4}$*-well-supported Nash equilibrium in an* n *players* $(2, 2)$*-SWLP games is PPAD-hard.*

The reduction consists of three stages. We first show the hardness result for the case of $(3, 2)$-sparse polymatrix game. In this stage, we follow the techniques of (Chen, Deng, and Teng 2006) for 2-player games and (Chen, Paparas, and Yannakakis 2017) for polymatrix games. Note that the above instance is not "win-lose", i.e., the payoff matrices contain entries that are neither "0"s nor "1"s. We refine the constructions of our gadgets to be "win-lose" ones, while keeping the resulting instances still $(3, 2)$-sparse. The final step is to fine-tune the GCIRCUIT, amortizing the number of "1"s in odd and even numbered rows of the payoff matrix $\mathbf{P}$, such that each row has at most two "1"s.

Make It Sparse

In the original proof of (Chen, Paparas, and Yannakakis 2017), the payoff matrix $\mathbf{P}$ is not that sparse due to two issues.

For the GCIRCUIT instance, some nodes serve as an input node of several gates simultaneously. In the original construction, this case will introduce many non-zero entries in the corresponding row of the resulting polymatrix game. Fortunately, it can be avoided easily by simulating the large out-degree nodes with a technical component "copy network" in (Chen, Deng, and Teng 2006). Formally, for each node v with out-degree at least three, we should invoke `COPY`$(\mathcal{T}; v)$ in Procedure 1. The sub-routine `INSERT`$(\mathcal{T}, T = (G, v_1, v_2, v_3, v, \alpha))$ adds a new gate T to the collection $\mathcal{T}$, and `REPLACE`(T, v, v') will replace the node v in the gate T with the node v'.

Lemma 3. *(Chen, Deng, and Teng 2006) [essentially]* GCIRCUIT *with max out-degree* 2 *is still PPAD-Complete.*

Procedure 1 `COPY`$(\mathcal{T}; v)$

```
1: let node $v$ be the input node of gates $T_1, \dots, T_l$.
2: Pick unused nodes $v_1, \dots, v_{l-1}$.
3: INSERT($\mathcal{T}, (G_=, v, \text{nil}, \text{nil}, v_1, \text{nil})$).
4: for all $i \in [l-2]$ do
5:     INSERT($\mathcal{T}, (G_=, v_i, \text{nil}, \text{nil}, v_{i+1}, \text{nil})$).
6: end for
7: for all $i \in [l-1]$ do
8:     REPLACE($T_{i+1}, v, v_i$).
9: end for
```

The other issue is about the constructions of game gadgets. The game gadgets for $G_\vee, G_\wedge, G_\zeta$ in the previous works are not column sparse. The remaining part is devoted to our modifications of these gadgets.

We introduce a new game gadget G_H to simplify the constructions of $G_\vee, G_\wedge, G_\zeta$. Informally, G_H is used to generate the constant $1/2$ approximately. To plug G_H into a gate $G \in \{G_\vee, G_\wedge, G_\zeta\}$, we extend the definition 3 by adding an auxiliary input node $v_3 \in V \cup \{\text{nil}\}$, i.e., $(G, v_1, v_2, v_3, v, \alpha)$, where v_3 is used to pass the constant $1/2$ as an input of the gate. To ensure the sparsity, for each $G_\vee, G_\wedge, G_\zeta$ type gate T, we need to pick an unused node u as the output node of a new G_H gate. We set u as the auxiliary input v_3 in T. For any other type of gates, the auxiliary input v_3 is set to nil.

All the game gadgets are summarized in Table 2. A special gate $G'_=$ will be defined and used in the following section, but we also put its game gadget here for convenience.

Next we prove that the resulting polymatrix game $\mathbf{P}$ satisfy all the constraints in the instance of GCIRCUIT listed in Table 2.

Lemma 4. *Given any ϵ-well-supported Nash equilibrium $(\mathbf{x}^*, \mathbf{y}^*)$ of the polymatrix game $(n, \mathbf{P})$, for any gate $T \in \mathcal{T}$, the construction in Table 1 satisfies the corresponding constraint in Table 2.*

Proof. Since each node cannot be taken as the output nodes of different gates, it is sufficient to show that each individual constraint is satisfied by our construction. Here we only prove the correctness of our newly constructed game gadget for G_H as an example. The correctness for other game gadgets follows similar argument, and one can refer to (Chen, Paparas, and Yannakakis 2017; Chen, Deng, and Teng 2006).

Matrices $\mathbf{L}[T]$, $\mathbf{R}[T]$, where $T = (G, v_1, v_2, v_3, v, \alpha)$
Initialize $\mathbf{L}[T], \mathbf{R}[T]$ as zero matrix.
Denote $L_{i,j} := \mathbf{L}[T]_{i,j}, R_{i,j} := \mathbf{R}[T]_{i,j}, k := C(v), k_1 := C(v_1), k_2 := C(v_2), k_3 := C(v_3)$

G_+ :	$\begin{cases} L_{2k-1,2k-1} = L_{2k,2k} = 1 \\ R_{2k_1-1,2k-1} = R_{2k_2-1,2k-1} = R_{2k-1,2k} = 1 \end{cases}$
G_- :	$\begin{cases} L_{2k-1,2k-1} = L_{2k,2k} = 1 \\ R_{2k_1-1,2k-1} = R_{2k_2-1,2k} = R_{2k-1,2k} = 1 \end{cases}$
$G_=$:	$\begin{cases} L_{2k-1,2k-1} = L_{2k,2k} = 1 \\ R_{2k_1-1,2k-1} = R_{2k-1,2k} = 1 \end{cases}$
$G_<$:	$\begin{cases} L_{2k-1,2k} = L_{2k,2k-1} = 1 \\ R_{2k_1-1,2k-1} = R_{2k_2-1,2k} = 1 \end{cases}$
$G_\neg$:	$\begin{cases} L_{2k-1,2k} = L_{2k,2k-1} = 1 \\ R_{2k_1-1,2k-1} = R_{2k_1,2k} = 1 \end{cases}$
G_H :	$\begin{cases} L_{2k-1,2k} = L_{2k,2k-1} = 1 \\ R_{2k-1,2k-1} = R_{2k,2k} = 1 \end{cases}$
G_ζ :	$\begin{cases} L_{2k-1,2k} = L_{2k,2k-1} = 1 \\ R_{2k-1,2k-1} = 1/2, R_{2k_3-1,2k} = \alpha \end{cases}$
$G_{\times\zeta}$:	$\begin{cases} L_{2k-1,2k-1} = L_{2k,2k} = 1 \\ R_{2k-1,2k} = 1, R_{2k_1-1,2k-1} = \alpha \end{cases}$
$G_\vee$:	$\begin{cases} L_{2k-1,2k-1} = L_{2k,2k} = 1 \\ R_{2k_1-1,2k-1} = R_{2k_2-1,2k-1} = R_{2k_3-1,2k} = 1 \end{cases}$
$G_\wedge$:	$\begin{cases} L_{2k-1,2k-1} = L_{2k,2k} = 1 \\ R_{2k_3-1,2k} = 1 \\ R_{2k_1-1,2k-1} = R_{2k_2-1,2k-1} = \frac{1}{3} \end{cases}$
$G'_=$:	$\begin{cases} L_{2k-1,2k} = L_{2k,2k-1} = 1 \\ R_{2k_1,2k-1} = R_{2k,2k} = 1 \end{cases}$

Table 2: Polymatrix game gadgets for GCIRCUIT gates

Let's denote $T = (G, v_1, v_2, v_3, v, \alpha), k = C(v), k_1 = C(v_1), k_2 = C(V_2), k_3 = C(V_3)$. Let $(\mathbf{x}^*, \mathbf{y}^*) \in \mathbb{R}^{4K}$ be an ϵ-well-supported Nash equilibrium of polymatrix game $\mathbf{P}$, where $\mathbf{x}^*, \mathbf{y}^* \in \mathbb{R}^{2K}$ represent the strategy profiles of the first and the last K players respectively. Recall that we have the value of node $\mathbf{x}[u] = x^*{}_{2C(u)-1}$ for each node $u \in V$.

Given a gate G_H, we consider the difference Δ_k between the utility of two actions of the $(K+k)$-th player

$$\Delta_k := \mathbf{x}^{*T}(\mathbf{R}[T]_{2k-1} - \mathbf{R}[T]_{2k}) = x^*_{2k-1} - x^*_{2k}.$$

If $\Delta_k > \epsilon$, we have $y^*_{2k-1} = 1$ by the definition of ϵ-well-supported Nash equilibrium. It then follows that $x^*_{2k} = 1$, hence $x^*_{2k-1} = x^*_{2k} + \Delta_k > 1$, which is a contradiction! Now we assume $\Delta_k < -\epsilon$, then we have $y^*_{2k} = 1$. This leads to $x^*_{2k-1} = 1$, hence $x^*_{2k} = x^*_{2k-1} - \Delta_k > 1$. Finally, we have $|\Delta_k| \le \epsilon$, and $\mathbf{x}[v] = x^*_{2k-1} = (1 \pm \epsilon)/2$.

$\square$

After discussing the correctness of these new game gadgets, we move to show that the sparsity constraint is also satisfied under our construction.

Lemma 5. *The payoff matrix* $\mathbf{P}$ *constructed from* GCIRCUIT $(V, \mathcal{T})$ *by game gadgets in Table 1 is* $(3,2)$*-sparse.*

Proof. Recall that $\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{A} & \mathbf{0} \end{pmatrix}$, and $\mathbf{A} = \sum_{T \in \mathcal{T}} \mathbf{L}[T]$, $\mathbf{B} = \sum_{T \in \mathcal{T}} \mathbf{R}[T]$. We only need to show the sparsity of $\mathbf{A}, \mathbf{B}$ respectively.

For each index $k = C(v) \in [K]$, where $v \in V$, there is at most one gate $T_v \in \mathcal{T}$ such that v is the output node of T. Also note that the only possible non-zero entries in rows and columns $2k-1, 2k$ of $\mathbf{A}$ is due to $\mathbf{L}[T_v]$, that contains exactly two "1"s on two different rows and columns in the (k,k)-th 2×2 block. Thus, $\mathbf{A}$ is $(1,1)$-sparse.

Next we consider the matrix $\mathbf{B}$. The only possible non-zero entries in columns $2k-1, 2k$ of $\mathbf{B}$ are due to $\mathbf{R}[T_v]$, which contains at most two "1"s in each column $2k-1, 2k$. For the rows $2k-1, 2k$, recall that v has out-degree at most two due to the "copy network", so v appears in at most two gates T_1, T_2 as input nodes and one gate T_v as the output node. Each of $\mathbf{R}[T_1], \mathbf{R}[T_2], \mathbf{R}[T_v]$ adds at most one "1" in rows $2k-1, 2k$. Therefore, $\mathbf{B}$ is $(3,2)$-sparse. Hence the lemma holds.

□

Make It "Win-Lose"

The gadgets for $G_\wedge, G_\zeta, G_{\times\zeta}$ gates in the previous step contain entries that are not zeros or ones. In this section, we simulate these gates with others which contain only zeros and ones.

First we can replace the AND gate $G_\wedge$ with three $G_\neg$ gates and one $G_\vee$ gate by De Morgan's Laws, i.e., $x \wedge y = \neg((\neg x) \vee (\neg y))$ for any Boolean values x and y. One can see that the error of the new AND gate is still ϵ, since the values of nodes used here are all Boolean.

For the gate $G_{\times\zeta}$, we have the following observation, showing that a simpler version of $G^*_{\times 0.5}$, which outputs nearly half of its first input, is enough for our use.

Observation 1. *During the reduction from* BROUWER *to* GCIRCUIT *in (Chen, Deng, and Teng 2009), establishing the PPAD-hardness of* GCIRCUIT *problem, the constant* α *in* $G_{\times\zeta}$ *only needs to be one of* $\{1/2, 1/4, 1/8\}$.

Hence, we can implement $G_{\times\zeta}$ in the reduction from BROUWER to GCIRCUIT with at most three gates $G^*_{\times 0.5}$.

Procedure 2 HALF$(\mathcal{T}; v; v_1)$

```
1: Pick an unused node $v_2 \in V$.
2: INSERT$(\mathcal{T}, (G_-, v, v_2, \text{nil}, v_1, \text{nil}))$.
3: INSERT$(\mathcal{T}, (G_=, v_1, \text{nil}, \text{nil}, v_2, \text{nil}))$.
```

For convenience, we construct the gate $G^*_{\times 0.5}$ with the combination of one gate G_- and one gate $G_=$, as the subroutine HALF listed in Procedure 2. In detail, if we want to let $\mathbf{x}[v_1] = \mathbf{x}[v]/2 \pm \epsilon$, we could use one additional unused node v_2, such that $\mathbf{x}[v_1] = \max(0, \mathbf{x}[v] - \mathbf{x}[v_2]) \pm \epsilon$. We next use a $G_=$ gate to enforce that $\mathbf{x}[v_2] = \mathbf{x}[v_1] \pm \epsilon$. It's straightforward to verify the following lemma.

Lemma 6. $G^*_{\times 0.5}$ *is implemented by* HALF *with error* ϵ.

For the gates $G_{\times\zeta}$ where $\zeta = 1/4, 1/8$, the errors are accumulated to $2\epsilon, 3\epsilon$ respectively.

The last one is G_ζ gate, which could be simulated by G_+ and $G^*_{\times 0.5}$ gates, based on the following observation.

Observation 2. *During the reduction from* BROUWER *to* GCIRCUIT *(Chen, Deng, and Teng 2009), establishing the PPAD-hardness of* GCIRCUIT *problem, the constants* α *in* G_ζ *only have values in* $\{1/2, 1/4, 1/8\} \cup \{k/(8K) : k \in [N^3]\}$*, where* N *is a parameter in the* BROUWER*, satisfying that* $K \geq N^{12}$ *and* K *is a power of 2.*

We construct a sub-routine GENCONST$(\mathcal{T}; v; v_h k; t)$ in Procedure 3, which can generate any constant $\alpha = k/2^t$ given any positive integers k and t.

Procedure 3 GENCONST$(\mathcal{T}; v; k; t)$

```
1: Pick unused node $v_1, \ldots, v_t, w_1, \ldots, w_k \in V$.
2: INSERT$(\mathcal{T}, (G_H, \text{nil}, \text{nil}, \text{nil}, v_1, \text{nil}))$.
3: for $i$ from 1 to $t-1$ do
4:     HALF$(\mathcal{T}, v_i, v_{i+1})$.
5: end for
6: INSERT$(\mathcal{T}, (G_=, v_t, \text{nil}, \text{nil}, w_1, \text{nil}))$.
7: for $i$ from 1 to $k-1$ do
8:     INSERT$(\mathcal{T}, (G_+, w_i, v_t, \text{nil}, w_{i+1}, \text{nil}))$.
9: end for
```

Now all game gadgets are of value zeros and ones. Recall that we have $K \geq N^{12}$, and by Observation 2, GENCONST needs $O(N^3)$ unused nodes. Hence we have enough unused nodes.

Notice that GENCONST may incur a large out-degree of the agent v_t, we still need to apply the "copy-network" structure after replacing $G_\zeta, G_{\times\zeta}$ and $G_\wedge$, to make sure that the out-degree of any node is at most two. In the construction, we keep all new game gadgets in Table 2 and only apply old gadgets to simulate all non-0/1 gates, so by the same argument in Lemma 5, we can show that $\mathbf{P}$ is still $(3,2)$-sparse.

Reduction to $(2,2)$-SWLP Games

Towards an optimal upper bound for the non-zero entries in each row, we need a more elaborate analysis. Intuitively, it's not difficult to notice that the number of "1"s appearing in rows $2k-1$ is more than "1"s appearing in rows $2k$ for any $k = C(v), v \in V$ in the most of the game gadgets summarized in Table 2. Specifically, we have the following observation.

Observation 3. *Let* $v \in V$ *be a node in the* GCIRCUIT *and* $k = C(v)$*, we have:*

1. *If* v *is the output node of a gate* $T \in \mathcal{T}$*, then row* $2k$ *is an all-zeros vector in* $\mathbf{R}[T]$*, unless the gate of* T *is* G_H.
2. *If* v *is one of the input nodes of a gate* $T \in \mathcal{T}$*, then row* $2k$ *is all-zeros vector in* $\mathbf{R}[T]$*, unless the gate of* T *is* $G_\neg$.

Algorithm 4 Balancing Nonzero Entries

```
Input: GCIRCUIT instance $(V, \mathcal{T})$.
1: for all node $v$ with out-degree 2 do
2:     Pick two unused nodes $v', v''$.
3:     INSERT$(\mathcal{T}, (G'_{=}, v, \text{nil}, \text{nil}, v', \text{nil}))$.
4:     INSERT$(\mathcal{T}, (G'_{=}, v, \text{nil}, \text{nil}, v'', \text{nil}))$.
5:     Let $T_1, T_2$ be the two gates that $v$ is one of their input
       nodes.
6:     REPLACE$(T_1, v, v')$.
7:     REPLACE$(T_2, v, v'')$.
8: end for
9: return $\mathcal{T}$
```

3. *No single gadget for any gate has at least two "1"s on one row.*

The observation above suggests that the distribution of "1" is very unbalanced between odd and even numbered rows. Therefore, we propose a new type of gate $G'_{=}$, which is equivalent to $G_{=}$, with a slightly different game gadget listed in the Table 2. Both input and output nodes v, v_1 of a $G'_{=}$ gate will have one "1" in row $2k, 2k_1$ instead of row $2k-1, 2k_1-1$ respectively, where $k = C(v), k_1 = C(v_1)$. Intuitively, $G'_{=}$ could be used to transport the "1" from the "crowded" odd number row to the "spare" even number row. The detailed process is summarized in the Algorithm 4 below.

Lemma 7. *By applying the Algorithm 4, the payoff matrix* $\mathbf{P}$ *is* $(2, 2)$*-sparse.*

Proof. Let $v \in V$ be a node in the GCIRCUIT and $k = C(v)$. We make a case analysis for the position of v in the instance of GCIRCUIT.

1. If v is not an output node of any gate, then rows $2k-1, 2k$ have at most two "1"s due to the out-degree limit.
2. If v is an output node of G_H or $G'_{=}$, it serves as the input node for at most one gate, guaranteed by the previous construction in Algorithm 4. Thus, rows $2k-1, 2k$ still have at most two "1"s there.
3. If v is an output node of any other type of gate, by the item 1 of Observation 3, row $2k$ has at most two "1"s. Algorithm 4 only inserts new $G'_{=}$ gates, so v cannot be a newly inserted node, and it should have been processed by Algorithm 4. Therefore, the row $2k-1$ will also have at most two "1"s in it.

$\square$

Remark 1. Even though our constructions in this section could accumulate errors, like copy networks and the subroutine GENCONST, the total error is accumulated *linearly* in terms of the number of gates for the GCIRCUIT instance $(V, \mathcal{T})$.

Recall that $|\mathcal{T}| \le |V| = K$, and $\epsilon = 1/K^3$ for the original proof about the PPAD-hardness result of the GCIRCUIT problem. Thus we could take $\epsilon' = 1/n^4 = 1/(2K)^4$ in the problem of calculating ϵ'-approximate well-supported Nash equilibrium in $(2, 2)$-sparse win-lose polymatrix game, and use it to recover an $K\epsilon' < \epsilon$ approximate solution of the GCIRCUIT problem.

Algorithm 5 Very Sparse Win-Lose Polymatrix Game

```
Input: Number of players $n$, the payoff matrix $\mathbf{P}$.
1: Initialize $r[i] \leftarrow 0, \forall i \in [2n]$, $S \leftarrow [n]$.
2: flag $\leftarrow$ true
3: while flag is true do
4:     flag $\leftarrow$ false
5:     for all $l \in [2n]$ do
6:         Suppose $l$ is one of actions of player $v$.
7:         if $v \notin S$ then
8:             continue
9:         end if
10:        Denote $l'$ as the other action of player $v$.
11:        if $r[l] + \text{Count}(\mathbf{P}, l) \le r[l']$ then
12:            /* $l$ is dominated by $l'$ */
13:            flag $\leftarrow$ true
14:            $x^*_l \leftarrow 0, x^*_{l'} \leftarrow 1$
15:            for all $j \in [2n], \mathbf{P}[l'][j] = 1$ do
16:                $r[j] \leftarrow r[j] + 1$
17:            end for
18:            $S \leftarrow S \backslash \{v\}$, clear the rows and the columns of
               player $v$ in $\mathbf{P}$ as zero.
19:        end if
20:    end for
21: end while
22: for all player $v \in S$ do
23:     Let $l, l'$ be the two actions of player $v$.
24:     $x^*_l, x^*_{l'} \leftarrow \frac{1}{2}$
25: end for
26: return $\mathbf{x}^*$
```

Algorithm

To complement our hardness result, we propose a simple and efficient algorithm inspired by (Chen, Deng, and Teng 2006) for both $(1, n)$ and $(n, 1)$-SWLP games.

The algorithm maintains a set S of players whose strategy hasn't been decided yet, the payoff matrix $\mathbf{P}$, the array $r[l]$, and the partial strategy profile $\mathbf{x}^*$. In which, $r[l]$ is the minimum utility for choosing action i respect to partial strategy profile $\mathbf{x}^*$.

In each round, the algorithm looks for the weakly dominant strategy. Specifically, let $\text{Count}(\mathbf{P}, l)$ be the subprocedure which counts the number of "1"s in the column l of $\mathbf{P}$. If l, l' are the two actions of a player $v \in S$ and $r[l] + \text{Count}(\mathbf{P}, l) \le r[l']$, then action l is dominated by action l'. We could now safely let player v choose l' with probability 1, remove v from S, update the array r, and clear the rows and the columns of player v in $\mathbf{P}$ as zero.

When there is no dominant strategy left, the algorithm will set probability $\frac{1}{2}$ to each of the actions of any player $v \in S$. The details are formally summarized in Algorithm 5.

Time Complexity. One can easily check that the time complexity of Algorithm 5 is bounded by $O(n^3)$. By using a better implementation for function "Count", we could further optimize the total running time to $O(n^2)$.

Analysis for the $(n,1)$-SWLP Case

When reaching the line 21 in Algorithm 5, for any player $v \notin S$, its strategy must be the best response under strategy profile $\mathbf{x}^*$, because she chooses the dominant strategy during the Algorithm 5. For any other player $v \in S$, we have the following claim.

Claim 1. *Suppose line 21 in Algorithm 5 has just been executed, then for any player $v \in S$, we have* $\mathrm{Count}(\mathbf{P}, l) = \mathrm{Count}(\mathbf{P}, l') = 1$*, where l, l' are the two actions of v.*

Proof. Recall the definition of $(n,1)$-sparse matrix, we know that $\mathrm{Count}(\mathbf{P}, l) \leq 1$ for any $l \in [2n]$. Thus, we have three possible cases to consider:

1. If $\mathrm{Count}(\mathbf{P}, l) = \mathrm{Count}(\mathbf{P}, l') = 0$, then there is no uncertainties left. We could know which of l, l' is the dominated strategy by comparing $r[l], r[l']$ in line 11 of Algorithm 5. Thus v is **not** in S in this case.
2. If $\mathrm{Count}(\mathbf{P}, l) = 0, \mathrm{Count}(\mathbf{P}, l') = 1$, we could still know which of l, l' is the dominated in line 11 of Algorithm 5. More specifically, if $1 + r[l'] \leq r[l]$, then l' is dominated. Otherwise we have $0 + r[l] \leq r[l']$, then l is dominated. Thus v is **not** in S in this case.
3. If $\mathrm{Count}(\mathbf{P}, l) = \mathrm{Count}(\mathbf{P}, l') = 1$, we must have $r[l] = r[l'] = 0$. This case is possible.

□

With the claim above, we know that the utility of any action of any player $v \in S$ is $\frac{1}{2}$, thus any player $v \in S$ also chooses its best response, and $\mathbf{x}^*$ is indeed a Nash equilibrium.

Analysis for the $(1,n)$-SWLP Case

The correctness of the player $v \notin S$ is the same. We now have the condition that each row has at most one of "1" instead, and we could get a generalization of Claim 1 in this case.

Claim 2. *Suppose reaching line 21 in Algorithm 5, then for any player $v \in S$, we have the following two possibilities:*

1. $\mathrm{Count}(\mathbf{P}, l) = \mathrm{Count}(\mathbf{P}, l') = 1$ *and* $r[l] = r[l']$*;*

2. $\mathrm{Count}(\mathbf{P}, l) = 0, \mathrm{Count}(\mathbf{P}, l') = 2$ *and* $r[l] = r[l'] + 1$*,*

where l, l' are the actions of v.

Proof. With the same argument as in the proof of Claim 1, we could rule out the case $\mathrm{Count}(\mathbf{P}, l) = 0, \mathrm{Count}(\mathbf{P}, l') = 0$ and $\mathrm{Count}(\mathbf{P}, l) = 0, \mathrm{Count}(\mathbf{P}, l') = 1$, i.e. we'll only have the case of $\mathrm{Count}(\mathbf{P}, l) + \mathrm{Count}(\mathbf{P}, l') \geq 2$.

By double-counting the "1"s in $\mathbf{P}$, we have the following inequality

$$2|S| \geq \sum_{u \in S} \mathrm{Count}(\mathbf{P}, l_u) + \mathrm{Count}(\mathbf{P}, l'_u) \geq 2|S|,$$

where l_u, l'_u are the actions of player u. The first inequality follows from the fact that each row have at most one of "1" and there are $2|S|$ rows haven't been cleared to zero in $\mathbf{P}$. Since both inequalities are actually tight, we must have $\mathrm{Count}(\mathbf{P}, l_u) + \mathrm{Count}(\mathbf{P}, l'_u) = 2$ for any player $u \in S$.

Now consider the relationship between $r[l]$ and $r[l']$ in this case. If $\mathrm{Count}(\mathbf{P}, l) = \mathrm{Count}(\mathbf{P}, l') = 1$, then $r[l] = r[l']$. Otherwise l or l' with smaller r will be dominated.

If $\mathrm{Count}(\mathbf{P}, l) = 0, \mathrm{Count}(\mathbf{P}, l') = 2$, then $r[l] = r[l'] + 1$. Otherwise l will be dominated if $r[l] < r[l'] + 1$ and l' will be dominated if $r[l] > r[l'] + 1$.

□

It's not difficult to check that in both cases of Claim 2, player $v \in S$ will have the same utility for both of its actions. Therefore $\mathbf{x}^*$ is indeed a Nash equilibrium.

Conclusions

In this paper, we study a very simple class of polymatrix games, Sparse Win-Lose Polymatrix (SWLP) games, whose payoff matrix is almost a zero matrix, with very few ones in each column and row. Our main result is a new dichotomy theorem for SWLP games: A PPAD-harness of equilibrium computation in $(2,2)$-SWLP games and an efficient algorithm of finding an exact Nash equilibrium in $(1,n)$ and $(n,1)$-SWLP games. The hardness result weakens the predictability of Nash equilibria in multi-player games.

Our result should also be of independent interest in complexity theory. A polymatrix game is the starting point in the complexity of market equilibrium and other succinct games. Where people used the hardness of polymatrix games before, one may try to use ours instead and may potentially get some new interesting results.

Rather than posing new open problems, we restate the problem of whether approximating a Nash equilibrium in *constant* Sparse Win-Lose Bimatrix games is tractable. It seems that we cannot bypass the barriers using our techniques, since all previous hardness results in bimatrix games need some base game enforcing that the probabilities of any two consecutive actions should be equal. In contrast, it is a natural property in polymatrix games.

Acknowledgements

This work is supported by Science and Technology Innovation 2030 –"New Generation Artificial Intelligence" Major Project (No. 2018AAA0100901), National Natural Science Foundation of China (No. 62002017) and Beijing Institute of Technology Research Fund Program for Young Scholars. We thank Yurong Chen for valuable discussions. Liu and Li contributed equally to this work. Zhengyang Liu is the corresponding author.

References

Addario-Berry, L.; Olver, N.; and Vetta, A. 2007. A Polynomial Time Algorithm for Finding Nash Equilibria in Planar Win-Lose Games. *J. Graph Algorithms Appl.* 11(1): 309–319. doi:10.7155/jgaa.00147.

Babichenko, Y.; Papadimitriou, C.; and Rubinstein, A. 2016. Can Almost Everybody Be Almost Happy? In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, 1–9. New York, NY, USA: Association for Computing Machinery.

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

Bilò, V.; and Mavronicolas, M. 2019. The Complexity of Computational Problems about Nash Equilibria in Symmetric Win-Lose Games. *CoRR* abs/1907.10468.

Chen, X.; and Deng, X. 2006. Settling the Complexity of Two-Player Nash Equilibrium. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 261–272.

Chen, X.; Deng, X.; and Teng, S.-H. 2006. Sparse games are hard. In *International Workshop on Internet and Network Economics*, 262–273. Springer.

Chen, X.; Deng, X.; and Teng, S.-H. 2009. Settling the complexity of computing two-player Nash equilibria. *J. ACM* 56(3).

Chen, X.; Durfee, D.; and Orfanou, A. 2015. On the Complexity of Nash Equilibria in Anonymous Games. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, 381–390. New York, NY, USA: Association for Computing Machinery.

Chen, X.; Paparas, D.; and Yannakakis, M. 2017. The Complexity of Non-Monotone Markets. *Journal of the ACM (JACM)* 64(3): 20.

Chen, X.; Teng, S.-H.; and Valiant, P. 2007. The Approximation Complexity of Win-Lose Games. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, 159–168. USA: Society for Industrial and Applied Mathematics.

Codenotti, B.; Leoncini, M.; and Resta, G. 2006. Efficient Computation of Nash Equilibria for Very Sparse Win-Lose Bimatrix Games. In Azar, Y.; and Erlebach, T., eds., *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, volume 4168 of *Lecture Notes in Computer Science*, 232–243. Springer. doi:10.1007/11841036_23.

Daskalakis, C.; Goldberg, P. W.; and Papadimitriou, C. H. 2009. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing* 39(1): 195–259.

Datta, S.; and Krishnamurthy, N. 2011. Some Tractable Win-Lose Games. In Ogihara, M.; and Tarui, J., eds., *Theory and Applications of Models of Computation - 8th Annual Conference, TAMC 2011, Tokyo, Japan, May 23-25, 2011. Proceedings*, volume 6648 of *Lecture Notes in Computer Science*, 365–376. Springer. doi:10.1007/978-3-642-20877-5_36.

Deligkas, A.; Fearnley, J.; and Savani, R. 2017. Computing constrained approximate equilibria in polymatrix games. In *International Symposium on Algorithmic Game Theory*, 93–105. Springer.

Deligkas, A.; Fearnley, J.; and Savani, R. 2020. Tree Polymatrix Games Are PPAD-Hard. In Czumaj, A.; Dawar, A.; and Merelli, E., eds., *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 38:1–38:14. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Deligkas, A.; Fearnley, J.; Savani, R.; and Spirakis, P. 2017. Computing approximate Nash equilibria in polymatrix games. *Algorithmica* 77(2): 487–514.

Garcia, C.; Lemke, C.; and Luethi, H. 1973. Simplicial Approximation of an Equilibrium Point for Non-Cooperative N-Person Games. In *Mathematical Programming*, 227–260. Academic Press.

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

Janovskaya, E. B. 1968. Equilibrium points in polymatrix games. *Latvian Mathematical Collection* .

Kuhn, H. W. 1961. An Algorithm for Equilibrium Points in Bimatrix Games. *Proceedings of the National Academy of Sciences of the United States of America* 47(10): 1657–1662.

Lemke, C. E.; and J. T. Howson, J. 1964. Equilibrium Points of Bimatrix Games. *Journal of the Society for Industrial and Applied Mathematics* 12(2): 413–423.

Lipton, R. J.; Markakis, E.; and Mehta, A. 2003. Playing Large Games Using Simple Strategies. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, EC '03, 36–41. New York, NY, USA: Association for Computing Machinery.

Liu, Z.; and Sheng, Y. 2018. On the Approximation of Nash Equilibria in Sparse Win-Lose Games. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, 1154–1160. AAAI Press.

Nash, J. 1951. Non-cooperative Games. *Annals of Mathematics* 54(2): 286–295.

Nash, J. F. 1950. Equilibrium points in n-person games. *Proc. of the National Academy of Sciences* 36: 48–49.

Ortiz, L. E.; and Irfan, M. T. 2017. Tractable Algorithms for Approximate Nash Equilibria in Generalized Graphical Games with Tree Structure. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, 635–641. AAAI Press.

Papadimitriou, C. H. 1994. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *J. Comput. Syst. Sci.* 48(3): 498–532.

Rubinstein, A. 2015. Inapproximability of Nash equilibrium. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, 409–418. ACM.

Savani, R.; and von Stengel, B. 2006. Hard-to-Solve Bimatrix Games. *Econometrica* 74(2): 397–429.

Shapley, L. S. 1974. *A note on the Lemke-Howson algorithm*, 175–189. Berlin, Heidelberg: Springer Berlin Heidelberg.

Wilson, R. 1971. Computing equilibria of n-person games. *SIAM Journal on Applied Mathematics* 21(1): 80–87.