

Parameterized Two-Player Nash Equilibrium

Danny Hermelin, Chien-Chung Huang, Stefan Kratsch, and Magnus Wahlström

Max-Planck-Institute for Informatics, Saarbrücken, Germany
{hermelin,villars,skratsch,wahl}@mpi-inf.mpg.de

Abstract. We study the problem of computing Nash equilibria in a two-player normal form (bimatrix) game from the perspective of parameterized complexity. Recent results proved hardness for a number of variants, when parameterized by the support size. We complement those results, by identifying three cases in which the problem becomes fixed-parameter tractable. Our results are based on a graph-theoretic representation of a bimatrix game, and on applying graph-theoretic tools on this representation.

1 Introduction

Algorithmic game theory is a quite recent yet rapidly developing discipline that lies at the intersection of computer science and game theory. The emergence of the internet has given rise to numerous applications in this area such as online auctions, online advertising, and search engine page ranking, where humans and computers interact with each other as selfish agents negotiating to maximize their own payoff utilities. The amount of research spent in attempting to devise computational models and algorithms for studying these types of interactions has been overwhelming in recent years; unsurprisingly perhaps, when one considers the economical rewards available in this venture.

The central problem in algorithmic game theory is that of computing a *Nash equilibrium*, a set of strategies for each player in a given game, where no player can gain by changing his strategy when all other players strategies remain fixed. This problem is so important because Nash equilibria provide a good way to predict the outcomes of many of the scenarios described above, and other scenarios as well. Furthermore, Nash's Theorem states that for any finite game a mixed Nash equilibrium always exists. However, for this concept to be meaningful for predicting behaviors of rational agents which are in many cases computers, a natural prerequisite is for it to be computable. This led researchers such as Papadimitriou to dub the problem of computing Nash equilibria as one of the most important complexity problems of our time [24].

The initial breakthrough in determining the complexity of computing Nash equilibria was made by Daskalakis, Goldberg, and Papadimitriou [11, 20]. These two papers introduced a reduction technique which was used by the authors for showing that computing a Nash equilibrium in a four player game is PPAD-complete. Shortly afterwards, this hardness result was simultaneously extended to three player games by Daskalakis and Papadimitriou [15], and by Chen and

Deng [5]. The case of two player (bimatrix) games was finally cracked a year later by Chen and Deng [6], who proved it to be PPAD-complete. This implied that the existence of a polynomial-time algorithm for the core case of bimatrix games is unlikely.

Since the result of Chen and Deng [6], the focus on computing Nash equilibria in bimatrix games was directed either towards finding approximate Nash equilibria [3, 7–9, 12, 13, 22], or towards finding special cases where exact equilibria can be computed in polynomial time [2, 8, 10, 21, 22]. Nevertheless, for general bimatrix games the best known algorithm for computing either approximate or exact equilibria essentially tries all possibilities for the support of both players (the set of strategies played with non-zero probability). Once the support of both players is known, one can compute a Nash equilibrium by solving a linear-program.

Theorem 1 ([23]). *A Nash equilibrium in a bimatrix game, where the support sizes are bounded by k , can be computed in $n^{O(k)}$ time.*

Due to the central role that the algorithm of Theorem 1 plays in computing exact and approximate Nash equilibria, it is natural to ask whether one can improve on its running-time substantially. In particular, can we remove the dependency on the support size from the exponent? The standard framework for answering such questions is that of parameterized complexity theory [16, 19]. Estivill-Castro and Parsa initiated the study of computing Nash equilibria in this context [18]. They showed that when the support size is taken as a parameter, the problem is W[2]-hard even in certain restricted settings. The implication of their result is a negative answer to the above question. In particular, combining their reduction with the results of Chen *et al.* [4] gives a sharp contrast to Theorem 1 above.

Theorem 2 ([18]). *Unless $\text{FPT}=\text{W}[1]$, there is no $n^{o(k)}$ time algorithm for computing a Nash equilibrium with support size at most k in a bimatrix game.*

The consequence of Theorem 2 above is devastating in the sense that for large enough games that have equilibria with reasonably small supports, the task of computing equilibria already becomes infeasible. The main motivation of this paper is to find scenarios where one can circumvent this. Our goal is thus to identify natural parameters which govern the complexity of computing Nash equilibria, and which can help in devising feasible algorithms. We believe that this direction can prove to be fruitful in the quest for understanding the computational limitations of this fundamental problem. Indeed, prior to our work, Kalyanaraman and Umans [21] provided a fixed-parameter algorithm for finding equilibrium in bimatrix games whose matrices have small rank (and some additional constraints).

Our techniques are based on considering a natural graph-theoretic representation of bimatrix games. This is done by taking the union of the underlying boolean matrix of the two given payoff matrices, and considering this matrix as the biadjacency matrix of a bipartite graph. A similar approach was taken by [10], and in particular by [2] who considered games that have an underlying

planar graph structure. Our work complements both these results as will be explained further on, and further exemplifies the strength of a graph-theoretical approach when computing Nash equilibria in bimatrix games.

A natural class of games that has a convenient interpretation in the graph-theoretic context is the class of ℓ -sparse games [8, 10, 14]. Here each column and row in both payoff matrices of the game have at most ℓ non-zero entries. An initial tempting approach in these types of games would be to try to devise a parameterized algorithm with ℓ taken as a single parameter. However, Chen, Deng, and Teng [8] showed that unless $\text{PPAD} = \text{P}$, there is no algorithm for computing an ε -approximate equilibrium for a 10-sparse game in time polynomial both in ε and n . Thus, such an FPT algorithm cannot exist unless PPAD is in P . We complement this result by showing that if ℓ is taken as a parameter, and the size of the supports is taken as an additional parameter, then computing Nash equilibrium is fixed-parameter tractable.

Theorem 3. *A Nash equilibrium in a ℓ -sparse bimatrix game, where the support sizes is bounded by k , can be computed in $\ell^{O(k\ell)} \cdot n^{O(1)}$ time.*

Note that the above result also complements the polynomial time algorithms given in [8, 10] for 2-sparse games. While in these algorithms there was no assumption made on the size of support of the equilibrium to be found, both algorithms could handle only *win-lose* games [1, 9], games with boolean payoff matrices. Theorem 3, on the other hand, holds for arbitrary payoffs.

Our second result is concerned with *k-unbalanced games*, games where the row player has a small set of k strategies [21, 22]. Lipton, Markakis, and Mehta [22] observed that in such games there is always an equilibrium where the row player plays a strategy with support size at most $k + 1$. Thus, by applying Theorem 1 one can find a Nash equilibrium in $n^{O(k)}$ time for these types of games. Can this result be improved to an algorithm running in $f(k) \cdot n^{O(1)}$ time? We give a partial answer to this question, by showing that if the number ℓ of different payoffs of the row player is taken as an additional parameter, the problem indeed becomes fixed-parameter tractable.

Theorem 4. *A Nash equilibrium in a k -unbalanced bimatrix game, where the row player has ℓ different payoff values, can be computed in $\ell^{O(k^2)} \cdot n^{O(1)}$ time.*

In our last result, examining the borderline of FPT cases, we consider a structural property that simultaneously extends both the previous cases (albeit only in the case of a bounded number of different payoffs). We show that for bimatrix games whose corresponding graph has locally bounded treewidth, and where the payoff matrices have at most ℓ different values, we can compute a Nash equilibrium of support size at most k in time $f(k, \ell) \cdot n^{O(1)}$. In addition to the above cases of sparse and unbalanced games, this also includes many other cases including games where the underlying graph structure is planar, as considered by [2]. However, as this class is quite general, the running-time dependency on both parameters is worse.

Theorem 5. *A Nash equilibrium in a locally bounded treewidth game, where the support sizes are bounded by k , and the payoff matrices have at most ℓ different values, can be computed in $f(k, \ell) \cdot n^{O(1)}$ time for some computable function $f()$.*

The paper is organized as follows: We begin with some preliminaries in Section 2. In Section 3 we consider ℓ -sparse games and prove Theorem 3. Section 4 addresses locally bounded treewidth games and proves Theorem 5. Finally, in Section 5 we prove Theorem 4 regarding k -unbalanced games.

2 Preliminaries

Let $\mathcal{G} := (A, B)$ be a bimatrix game, where $A, B \in \mathbb{Q}^{n \times n}$ are the *payoff matrices* of the *row* and the *column* players respectively. The row (column) player has a strategy space consisting of the rows (columns) $[n] := \{1, \dots, n\}$. (For ease of notation, except in unbalanced games, we assume that both players have the same number of strategies; different numbers of strategies do not affect any of our results.) The row (column) player chooses a strategy profile x (*resp.* y), which is a probability distribution over his strategy space. That is, $x_i, y_j \geq 0$ for all $i, j \in [n]$, and furthermore $\sum_{i=1}^n x_i = 1$ and $\sum_{j=1}^n y_j = 1$. The expected outcomes of the game for the row and the column players are $x^T A y$ and $x^T B y$ respectively.

The players are rational, always aiming for maximizing their expected payoffs. They have reached a *Nash equilibrium* if the current strategies x and y are such that neither player has a deviating strategy \hat{x} or \hat{y} such that $\hat{x}^T A y > x^T A y$ or $x^T B \hat{y} > x^T B y$. In other words, if neither of them can improve his payoff independently of the other. The following proposition gives an equivalent condition for a pair of strategies to be an equilibrium.

Lemma 1. ([23, Chapter 3]) *The pair of strategy vectors (x, y) is a Nash equilibrium for the bimatrix game (A, B) if and only if*

- (i) $x_s > 0 \Rightarrow (A y)_s \geq (A y)_j$ for all $j \neq s$;
- (ii) $y_s > 0 \Rightarrow (x^T B)_s \geq (x^T B)_j$ for all $j \neq s$.

The *support* of a strategy vector x is defined as the set $S(x) = \{i : x_i > 0\}$. Note that the above proposition implies that if (x, y) is a Nash equilibrium, in the column vector $A y$, the entries in $S(x)$ are equivalent and no less than all other entries not in $S(x)$; symmetrically, in the row vector $x^T B$, the entries in $S(y)$ are equivalent and no less than other entries not in $S(y)$. It is known that, given possible supports $I, J \subseteq [n]$ it can be efficiently decided whether there is a matching Nash equilibrium, and the corresponding strategy vectors can be computed via linear programming.

The following graph associated with a bimatrix game is useful for presenting our algorithms in Sections 3 and 4.

Definition 1. Let $\mathcal{G} = (A, B)$ be a bimatrix game with $A, B \in \mathbb{Q}^{n \times n}$. The undirected bipartite graph $G := G(\mathcal{G})$ associated with \mathcal{G} is defined to be the bipartite graph with vertex classes $V_r := \{r_1, \dots, r_n\}$ and $V_c := \{c_1, \dots, c_n\}$, referred to as row resp. column vertices, where $r_i \in V_r$ and $c_j \in V_c$ are adjacent in G iff $A_{i,j} \neq 0$ or $B_{i,j} \neq 0$.

As a last bit of notation: For $I, J \subseteq [n]$, and any $n \times n$ matrix A , we use $A_{I,J}$ to denote the submatrix composed of rows in I and columns in J . We also use $A_{I,*}$ as a shorthand for $A_{I,[n]}$. Thus, $A_{i,*}$ means the i 'th row of A .

3 Sparse Games

In this section we present the proof for Theorem 3. Throughout the section we let $\mathcal{G} := (A, B)$ denote our given bimatrix game, where A and B are rational value matrices with at most ℓ non-zero entries per row or column. We will present an algorithm for finding a Nash equilibrium where the support sizes of both players are at most k (and k is taken as a parameter). The high-level strategy is to show that it suffices to search for equilibria that induce one or two connected components in the associated graph $G = G(\mathcal{G})$. This permits us to find candidate support sets by enumerating subgraphs of G (on one or two components). Central to this strategy is the notion of minimal equilibria:

Definition 2. A Nash equilibrium (x, y) is minimal if for any Nash equilibrium (x', y') with $S(x') \subseteq S(x)$ and $S(y') \subseteq S(y)$, we have $S(x') = S(x)$ and $S(y') = S(y)$.

Our algorithm iterates through all possible support sizes $k_1, k_2 \leq k$ in increasing order to determine whether there exists an equilibrium (x, y) with $|S(x)| = k_1$ and $|S(y)| = k_2$. To avoid cumbersome notation, we will assume that $k_1 = k_2 = k$ (extending this to general case will be immediate). Thus at a given iteration, the algorithm can assume that no equilibrium exists with smaller supports, which means it can restrict its search to minimal equilibria. This fact will prove crucial later on. Furthermore, observe that we can assume $n > \ell k$ since otherwise obtaining the running time required by Theorem 3 is trivial using Theorem 1. Therefore, since our game is ℓ -sparse, our algorithm only needs to search for equilibria where both players receive non-negative payoffs.

Lemma 2. If $\mathcal{G} = (A, B)$ is an ℓ -sparse game, where $A, B \in \mathbb{Q}^{n \times n}$ and $n > \ell k$, then in any Nash equilibrium with support at most $k \times k$, both players receive non-negative payoffs.

For an equilibrium (x, y) , let the *extended support* of x be the rows $S(x) \cup N(S(y))$, and similarly for y , where the neighborhood $N(I)$ is taken over the graph $G := G(\mathcal{G})$ of the game. Note that any row not in the extended support of x would have payoff constantly zero given the current strategy of y , and thus is not important for the existence of an equilibrium. We will show that for a minimal equilibrium (x, y) , the extended supports of x and y induce a subgraph

of G which has at most two connected components. This will be done in two steps: The first is the special case where $A_{S(x),S(y)} = B_{S(x),S(y)} = \mathbf{0}$, while the second corresponds to the remaining cases.

Lemma 3. *If (x, y) is a minimal Nash equilibrium for a game (A, B) with $A_{S(x),S(y)} = B_{S(x),S(y)} = \mathbf{0}$, then the subgraphs induced by $N[S(x)]$ and $N[S(y)]$ in the graph associated with the game are both connected.*

Proof. Let $H := G[N[S(x)]]$ be the subgraph of G induced by $N[S(x)]$, and aiming towards a contradiction, suppose that H is disconnected. Let C be a connected component in H , and write $p := \sum_{i \in V_r(C)} x_i$ to denote the probability that a row strategy in C is played according to x . Now define a new row strategy by setting $\hat{x}_i = x_i/p$ if $i \in V_r(C)$, and $\hat{x}_i = 0$ otherwise. We argue that (\hat{x}, y) is a Nash equilibrium, contradicting the fact that (x, y) is minimal.

Obviously, the expected payoff in (\hat{x}, y) is zero for both players, as $A_{S(\hat{x}),S(y)} = B_{S(\hat{x}),S(y)} = \mathbf{0}$. Furthermore, there is no row i such that $(Ay)_i > 0$, since the strategy y is unchanged and the original strategy pair (x, y) is an equilibrium. Now assume that there is a column j such that $(\hat{x}^T B)_j > 0$. Then $B_{i,j} \neq 0$ for some $i \in S(\hat{x})$, and by the connectivity assumption we must have $B_{i,j} = 0$ for all $i \in S(x) \setminus S(\hat{x})$, but then $(x^T B)_j = p(\hat{x}^T B)_j > 0$, which contradicts the assumption that (x, y) is a Nash equilibrium. By Lemma 1, it follows that (\hat{x}, y) is a Nash equilibrium. \square

Lemma 4. *If (x, y) is a minimal Nash equilibrium for a game (A, B) with either $A_{S(x),S(y)} \neq \mathbf{0}$ or $B_{S(x),S(y)} \neq \mathbf{0}$, and with a non-negative payoff for both players, then the subgraph induced by $N[S(x) \cup S(y)]$ is connected.*

Proof. Let H be the subgraph induced by $N[S(x) \cup S(y)]$ and suppose that H is disconnected. We will derive a contradiction by showing that (x, y) is not minimal.

Let C be a connected component in H intersecting both $S(x)$ and $S(y)$, and let $V_{\hat{x}} := V_r(C) \cap S(x)$ and $V_{\hat{y}} := V_c(C) \cap S(y)$ denote the set of row and column strategies in C , respectively. We define a new pair of strategy profiles (\hat{x}, \hat{y}) where $S(\hat{x}) = V_{\hat{x}}$ and $S(\hat{y}) = V_{\hat{y}}$, by normalizing (x, y) onto $V_{\hat{x}}$ and $V_{\hat{y}}$. That is, we let $p := \sum_{i \in V_{\hat{x}}} x_i$, and set $\hat{x}_i = x_i/p$ if $i \in V_{\hat{x}}$, and $\hat{x}_i = 0$ otherwise. Similarly, we let $q := \sum_{j \in V_{\hat{y}}} y_j$, and set \hat{y}_j accordingly. As either $S(\hat{x}) \subset S(x)$ or $S(\hat{y}) \subset S(y)$, to prove the lemma it suffices to argue that (\hat{x}, \hat{y}) is an equilibrium.

Consider a row strategy $i \in [n]$. We claim that

$$(A\hat{y})_i = \begin{cases} (Ay)_i/q & \text{if } A_{i,V_{\hat{y}}} \neq \mathbf{0}, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The second case is clear. For the first case, assume that $A_{i,V_{\hat{y}}} \neq \mathbf{0}$. Now, if $A_{i,j} \neq 0$ for $j \in S(y) \setminus V_{\hat{y}}$, then there would be an edge in H from the vertex corresponding to row i , which is in C , to the vertex corresponding to column j , which is not in C , contradicting that C is a connected component.

Thus

$$(Ay)_i = \sum_{j \in S(y)} A_{i,j} y_j = \sum_{j \in V_{\hat{y}}} A_{i,j} (q\hat{y}_j) + \sum_{j \in S(y) \setminus V_{\hat{y}}} A_{i,j} y_j = q(A\hat{y})_i + 0$$

and the claim follows.

Now let $s \in S(\hat{x})$, and consider some arbitrary row strategy $i \in [n]$. Assume by way of contradiction that $(A\hat{y})_s < (A\hat{y})_i$. It is clear that $A_{i,V_{\hat{y}}} \neq \mathbf{0}$, thus $(A\hat{y})_i = (Ay)_i/q$; we consider the cases for row s . If $A_{s,V_{\hat{y}}} \neq \mathbf{0}$, then by (1), we have $(A\hat{y})_s = (Ay)_s/q$, implying $(Ay)_s < (Ay)_i$. On the other hand, if $A_{s,V_{\hat{y}}} = \mathbf{0}$ but $A_{s,S(y)} \neq \mathbf{0}$, then C would not be a connected component of H (since a neighbor of s would be missing). Thus $A_{s,S(y)} = \mathbf{0}$ and $(Ay)_s = (A\hat{y})_s = 0$, and $(Ay)_i = q(A\hat{y})_i > 0$. In both cases we contradict that (x, y) is an equilibrium. Thus we fulfill condition (i) of Lemma 1, and by symmetry we also fulfill condition (ii). We have shown that (\hat{x}, \hat{y}) is an equilibrium, contradicting the minimality of (x, y) . \square

As an immediate corollary of Lemma 3 and 4, we get that the subgraph in $G(\mathcal{G})$ induced by the extended support of a minimal equilibrium has at most two connected components. In the following lemma we show that in graphs of small maximum degree, we can find all such subgraphs quite efficiently. This will allow us to find a small, minimal equilibrium by checking all sets of rows and columns that would be candidates for being the extended supports of one.

Lemma 5. *Let G be a graph on n vertices and with maximum degree $\Delta = \Delta(G)$. In time $(\Delta + 1)^{2t} \cdot n^{c+O(1)}$ one can enumerate all subgraphs on t vertices that consist of c connected components.*

Proof. We first show how to enumerate all connected subgraphs on t vertices in time $(\Delta + 1)^{2t} \cdot n^{O(1)}$ by a branching algorithm. At any point selected vertices will be active or passive. When a vertex is selected it will first be active and later be set to passive. Selecting a vertex and making it active respectively setting a vertex to passive is called an *event*.

First, we branch on the choice of one out of n starting vertices in G and set it active. Then until we have selected t vertices we do the following: We consider the least recently added active vertex and branch on one of at most $\Delta + 1$ events, namely selecting one of its at most Δ neighbors (and making it active) or setting the vertex itself to passive. We terminate when we have selected t vertices (and output the corresponding subgraph) or when there are no more active vertices. Clearly, on each branch of this algorithm there are at most $2t$ events. Thus the branching tree has at most $(\Delta + 1)^{2t}$ leaves, implying a total runtime of $(\Delta + 1)^{2t} \cdot n^{O(1)}$. Observe that for every connected subgraph on t vertices there is a sequence of events such that the graph occurs in the enumeration.

The generalization to c components is straightforward: When there are no more active vertices but we have not yet selected c components, then we select one of the less than n remaining vertices of G as a new active vertex, starting a new component. Selecting new starting vertices adds a factor of n^c to our runtime. \square

We are now in position to describe our entire algorithm. It first iterates through all possible sizes of extended support in increasing order. In each iteration, it enumerates all subgraphs that might correspond to the extended support of a minimal equilibrium. It then checks all ways of selecting a support from the given subgraph, and for each such selection it uses the algorithm behind Theorem 1 to check whether there is an equilibrium on the support. If no equilibrium is found throughout the whole process, the algorithm reports that there exists no equilibrium with support size at most k in \mathcal{G} . The running time is bounded by $\ell^{O(k\ell)} n^{O(1)}$ from Lemma 5, times $\binom{k\ell+k}{k}^2 = 2^{O(k\ell)}$ ways of selecting the support, times $n^{O(1)}$ for checking for an equilibrium. In total, we get a running time of $\ell^{O(k\ell)} n^{O(1)}$.

Finally, completeness comes from the exhaustiveness of Lemma 5 and the structure given by Lemmas 3 and 4.

3.1 Non-negative payoffs

In the case that the payoffs of our games are non-negative, i.e., $A, B \in \mathbb{Q}_{\geq 0}^{n \times n}$, we can reduce our running time to be polynomial in ℓ , for ℓ -sparse games. We begin with a strengthening of Lemmas 3 and 4.

Lemma 6. *Let $\mathcal{G} = (A, B)$ be a bimatrix game with $A, B \in \mathbb{Q}_{\geq 0}^{n \times n}$, and G be the graph associated with \mathcal{G} . If (x, y) is a minimal Nash equilibrium for \mathcal{G} , then either $|S(x)| = |S(y)| = 1$, or $G[S(x) \cup S(y)]$ is connected.*

Proof. Let $G_S := G[S(x) \cup S(y)]$; assume that G_S is not connected. If the expected outcome is zero, then (since the entries are non-negative) every entry in $A_{S(x),*}$ and $B_{*,S(y)}$ is zero, and we get an equilibrium by selecting any single row $i \in S(x)$ and column $j \in S(y)$. Otherwise, every row of $A_{S(x),S(y)}$ and every column of $B_{S(x),S(y)}$ contains some positive entry. Let C be a connected component of G_S on row vertices $V_{\hat{x}}$ and column vertices $V_{\hat{y}}$, and define a new pair of strategy profiles (\hat{x}, \hat{y}) where $S(\hat{x}) = V_{\hat{x}}$ and $S(\hat{y}) = V_{\hat{y}}$, by normalizing (x, y) onto $V_{\hat{x}}$ and $V_{\hat{y}}$ as in Lemma 4. We will argue that (\hat{x}, \hat{y}) is an equilibrium.

Let $s \in V_{\hat{x}}$, and assume by way of contradiction that for some row $i \in [n]$, we have $(A\hat{y})_i > (A\hat{y})_s$. Let $(A\hat{y})_s = c_0$; by non-connectivity of G_S , $(Ay)_s = qc_0$. Further let $(A\hat{y})_i = c_1$ and $(Ay)_i = qc_1 + (1-q)c_2$. Now $(Ay)_s = qc_0 < qc_1 \leq qc_1 + (1-q)c_2 = (Ay)_i$, contradicting our assumptions; the last inequality is because the entries are non-negative. Repeating the argument symmetrically, we find that (\hat{x}, \hat{y}) is a Nash equilibrium. \square

Thus, to find an equilibrium in $\mathcal{G} = (A, B)$, it suffices to search for occurrences of the support, rather than the extended support. Invoking Lemma 5 directly with a bound of $2k$ vertices gives a running time of $\ell^{O(k)} n^{O(1)}$.

4 Locally Bounded Treewidth Games

Let $\mathcal{G} = (A, B)$ be a given game with $A, B \in P^{n \times n}$, with $P \subset \mathbb{Q}$, $|P| \leq \ell$, and let $G = G(\mathcal{G})$ the graph associated with \mathcal{G} . In this section we will present an

algorithm that finds an equilibrium with support sizes at most k when G comes from a graph class with locally bounded treewidth. Note that this is a partial extension of the results of the previous section, as graphs of bounded degree have locally bounded treewidth, while on the other hand we assume that there is a bounded set P of only ℓ different payoff values which can occur in the games. (The case $P = \{0, 1\}$ would correspond to win-lose games.)

Definition 3 ([17]). *A graph class has locally bounded treewidth if there is a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that for every graph $G := (V, E)$ of the class, any vertex $v \in V$, and any $d \in \mathbb{N}$, the subgraph of G induced by all vertices within distance at most d from v has treewidth at most $f(d)$.*

We refer readers to [19] for more details on the notion of treewidth and locally bounded treewidth. The crucial property of locally bounded treewidth graphs in our context is that first-order queries can be answered in FPT time on such graphs when the parameter is the size of first-order formula [19, Chapter 12.2].

For ease of presentation we show how to find an equilibrium where both players have support size k (the algorithm can be easily adapted to support sizes $k_1, k_2 \leq k$). Let I and J be two subsets of k elements in $[n]$. We say that two matrices $A^*, B^* \in \mathbb{Q}^{k \times k}$ occur in \mathcal{G} at (I, J) if $A^* = A_{I,J}$ and $B^* = B_{I,J}$. The pair (A^*, B^*) forms an *equilibrium pattern* if there exists an equilibrium (x, y) where (A^*, B^*) occur in \mathcal{G} at $(S(x), S(y))$. Our algorithm will try all possible ℓ^{2k^2} pairs of matrices (A^*, B^*) , and for each such pair it will determine whether it is an equilibrium pattern.

When does a pair of matrices (A^*, B^*) form an equilibrium pattern? The first obvious condition is that it occurs in \mathcal{G} at some pair of position sets (I, J) . Furthermore, by definition of a Nash equilibrium, there is a pair of strategies (x, y) with $S(x) = I$ and $S(y) = J$, such that neither player has a better alternative. The difficulty here lies in the fact that, even given the support $S(y)$ of the column player, there may be too many possible strategies for the row player that have supports different from I . To circumvent this, we define equivalence of rows with respect to supports $S(y)$, and of columns with respect to supports $S(x)$.

Definition 4. *Let $I, J \subseteq [n]$. Two rows $i_1, i_2 \in [n]$ are J -equivalent if $A_{i_1, J} = A_{i_2, J}$. Similarly, two columns $j_1, j_2 \in [n]$ are I -equivalent if $B_{I, j_1} = B_{I, j_2}$.*

Lemma 7. *Let J be the support of the column player. For any row strategy x there is a row strategy \hat{x} such that:*

- (i) *the support $S(\hat{x})$ contains at most one row from each J -equivalence class*
- (ii) *and for any column strategy y with support J we have $\hat{x}^T A y = x^T A y$.*

The same is true for column strategies, given a support I of the row player.

For each possible equilibrium pattern (A^*, B^*) we do the following. For each choice of rows $A^\dagger \subseteq P^{1 \times k}$ that do not occur in A^* and each choice of columns $B^\dagger \subseteq P^{k \times 1}$ that do not occur in B^* , we create two matrices

$$C = \begin{pmatrix} A^* & \mathbf{0} \\ A^\dagger & \mathbf{0} \end{pmatrix} \text{ and } D = \begin{pmatrix} B^* & B^\dagger \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

We then check if there is an equilibrium (x, y) in the game (C, D) with $S(x) = S(y) = [k]$ using linear programming. If there is such an equilibrium, then we proceed as follows to find an occurrence of (A^*, B^*) that avoids the rows and columns which were not chosen. For this let F_1 be the rows which occur neither in A^* nor in A^\dagger and let F_2 be the columns which occur neither in B^* nor in B^\dagger . We say that F_1 and F_2 are *forbidden* for (A^*, B^*) . We note that given (A^*, B^*) , a set of rows $F_1 \subseteq P^{1 \times k}$, and a set of columns $F_2 \subseteq P^{k \times 1}$, one can write a first-order formula of size bounded by some function in k and $|P| = \ell$ to determine whether (A^*, B^*) has an occurrence which avoids F_1 and F_2 .

Example 1. Consider a win-lose game (A, B) encoded into relations A and B such that $A(r, c)$ is true iff $A_{r,c} = 1$, and likewise for B . Then a 2×2 equilibrium where the pattern is two identity matrices, and the all one vector is a forbidden row and column, can be found with the formula

$$\begin{aligned} \exists r_1, r_2, c_1, c_2 & A(r_1, c_1) \wedge \neg A(r_1, c_2) \wedge \neg A(r_2, c_1) \wedge A(r_2, c_2) \wedge \\ & B(r_1, c_1) \wedge \neg B(r_1, c_2) \wedge \neg B(r_2, c_1) \wedge B(r_2, c_2) \wedge \\ & \forall r' (\neg A(r', c_1) \vee \neg A(r', c_2)) \wedge \forall c' (\neg B(r_1, c') \vee \neg B(r_2, c')). \end{aligned}$$

In general, with ℓ different values, there would be $\ell - 1$ relations A_i and B_i encoding the game, where $A_i(r, c)$ is true if $A_{r,c} = z_i$, for every $z_i \in P$ except the zero value.

Since the number of possible choices for F_1 and F_2 is bounded by some function in k and ℓ , and for each such choice we can determine whether F_1 and F_2 is a forbidden pair for (A^*, B^*) in polynomial-time, the total time for determining whether (A^*, B^*) is an equilibrium pattern is FPT in k and ℓ . Since the number of pairs (A^*, B^*) is also bounded by a function in k , the total running time of our entire algorithm is also FPT in k and ℓ .

To complete the proof of Theorem 5, let us briefly argue completeness. Assume that there is any equilibrium with support sizes equal to k , let I and J be the supports, and let A^* and B^* be corresponding sub-matrices. Observe that we may set all entries in columns outside J of A to zero without harm, ditto for rows outside I in B . According to Lemma 7 it suffices to keep one copy of each row outside A^* in A (also discard the corresponding zero-row in B to keep the size the same). The same is of course true for columns outside B^* in B . Except for a permutation this is equal to one of the games (C, D) that we considered. Therefore our algorithm will find such an equilibrium if one exists.

5 Unbalanced Games

In this section we briefly consider k -unbalanced bimatrix games. A bimatrix game (A, B) is k -unbalanced if $A, B \in \mathbb{Q}_{\geq 0}^{k \times n}$ for some $k \ll n$ [21, 22] (i.e., the row player has a significantly smaller number of strategies than the column player). We will show that a Nash equilibrium can be computed in FPT-time with respect to k and ℓ , where ℓ denotes the number of different payoffs that the row player has, i.e., $\ell := |\{A_{i,j} : 1 \leq i \leq k, 1 \leq j \leq n\}|$.

Similar to Definition 4 we define two column strategies $i, j \in [n]$ to be equivalent if $A_{*,i} = A_{*,j}$. (However, notice that unlike Definition 4, here equivalence of column strategies is defined with respect to the row player payoff.)

Lemma 8. *For each equilibrium there is an equilibrium where the column player plays at most one column from each equivalence class.*

Using Lemma 8 we can easily devise an FPT algorithm for computing a Nash equilibrium in our setting. The algorithm simply guesses the support of the row player and column player, and then uses the method of Theorem 1 to determine whether there exists a Nash equilibrium corresponding to these sets of supports. Observe that there are at most ℓ^k column-strategy equivalence classes. Furthermore, according to Lipton, Markakis, and Mehta [22], in a k -unbalanced game there always exists an equilibrium where the column player has support size at most $k + 1$. Thus the number of guesses the algorithm makes is bounded by $2^k \cdot \binom{\ell^k}{k+1} = \ell^{O(k^2)}$, and for each such guess, the amount of time required is polynomial. This completes the proof of Theorem 4.

References

1. T.G. Abbott, D.M. Kane, and P. Valiant. On the complexity of two-player win-lose games. In *Proc. of the 46th annual IEEE symposium on Foundations Of Computer Science (FOCS)*, pages 113–122, 2005.
2. L. Addario-Berry, N. Olver, and A. Vetta. A polynomial time algorithm for finding Nash equilibria in planar win-lose games. *Journal of Graph Algorithms and Applications*, 11(1):309–319, 2007.
3. H. Bosse, J. Byrka, and E. Markakis. New algorithms for approximate nash equilibria in bimatrix games. *Theoretical Computer Science*, 411(1):164–173, 2010.
4. J. Chen, B. Chor, M.R. Fellows, X. Huang, D.W. Juedes, I.A. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005.
5. X. Chen and X. Deng. 3-NASH is PPAD-complete. *Electronic Colloquium on Computational Complexity*, (134), 2005.
6. X. Chen and X. Deng. Settling the complexity of two-player nash equilibrium. In *Proc. of the 47th annual IEEE symposium on Foundations Of Computer Science (FOCS)*, pages 261–272, 2006.
7. X. Chen, X. Deng, and S.-H. Teng. Computing nash equilibria: Approximation and smoothed complexity. In *Proc. of the 47th annual IEEE symposium on Foundations Of Computer Science (FOCS)*, pages 603–612, 2006.
8. X. Chen, X. Deng, and S.-H. Teng. Sparse games are hard. In *Proc. of the 2nd international Workshop on Internet and Network Economics (WINE)*, pages 262–273, 2006.
9. X. Chen, S.-H. Teng, and P. Valiant. The approximation complexity of win-lose games. In *Proc. of the 18th annual ACM-SIAM Symposium On Discrete Algorithms (SODA)*, pages 159–168, 2007.
10. B. Codenotti, M. Leoncini, and G. Resta. Efficient computation of nash equilibria for very sparse win-lose bimatrix games. In *Proc. of the 14th Annual European Symposium on Algorithms (ESA)*, pages 232–243, 2006.

11. C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The complexity of computing a nash equilibrium. In *Proc. of the 38th annual ACM Symposium on Theory Of Computing (STOC)*, pages 71–78, 2006.
12. C. Daskalakis, A. Mehta, and C.H. Papadimitriou. Progress in approximate nash equilibria. In *Proc. of the 8th ACM Conference on Electronic Commerce (EC)*, pages 355–358, 2007.
13. C. Daskalakis, A. Mehta, and C.H. Papadimitriou. A note on approximate nash equilibria. *Theoretical Computer Science*, 410(17):1581–1588, 2009.
14. C. Daskalakis and C.H. Papadimitriou. On oblivious ptas’s for nash equilibrium. In *Proc. of the 41st annual ACM Symposium on Theory Of Computing (STOC)*, pages 75–84, 2009.
15. K. Daskalakis and C.H. Papadimitriou. Three-player games are hard. *Electronic Colloquium on Computational Complexity*, (139), 2005.
16. R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
17. D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3), 1999.
18. V. Estivill-Castro and M. Parsa. Computing nash equilibria gets harder: New results show hardness even for parameterized complexity. In *Proc. of the 15th Computing: the Australasian Theory Symposium (CATS)*, volume 94, pages 81–87, 2009.
19. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
20. P.W. Goldberg and C.H. Papadimitriou. Reducibility among equilibrium problems. In *Proc. of the 38th annual ACM Symposium on Theory Of Computing (STOC)*, pages 61–70, 2006.
21. S. Kalyanaraman and C. Umans. Algorithms for playing games with limited randomness. pages 323–334, 2007.
22. R.J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *Proc. of the 4th ACM Conference on Electronic Commerce (EC)*, pages 36–41, 2003.
23. N. Nisan, T. Roughgarden, É. Tardos, and V.V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.
24. C.H. Papadimitriou. Algorithms, games, and the internet. In *Proc. of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 749–753, 2001.

A Omitted proofs

A.1 Section 3

Proof (Lemma 2). Suppose (x, y) is an equilibrium with $|S(x)| \leq k$ and $|S(y)| \leq k$, where the row player has a negative payoff. Since the A is ℓ -sparse, and $n > \ell k$, there exists an all zero row $i_0 \in [n]$ in $A_{*, S(y)}$. Thus, letting the row player choose the strategy vector \hat{x} with $\hat{x}_{i_0} = 1$ and $\hat{x}_i = 0$ for all $i \neq i_0$, we get that $\hat{x}^T A y = 0 > x^T A y$, contradicting the fact that (x, y) is an equilibrium. The argument for the column player is symmetric. \square

A.2 Section 4

Proof (Lemma 7). We prove the lemma row strategies given a support J of the column player. The proof for column strategies is similar. Let x be a row strategy which includes two strategies i_1 and i_2 that are J -equivalent. It suffices to show that there is a row strategy \hat{x} with $S(\hat{x}) = S(x) \setminus i_2$ and $\hat{x}^T A y = x^T A y$ for any column strategy y with support J . For this, take \hat{x} to be the strategy defined by $\hat{x}_{i_1} := x_{i_1} + x_{i_2}$, $\hat{x}_{i_2} := 0$, and $\hat{x}_i = x_i$ for all $i \in \{1, \dots, n\} \setminus \{i_1, i_2\}$. Let y be any column strategy with $S(y) = J$. By definition of J -equivalence, we have $(A y)_{i_1} = (A y)_{i_2}$. Thus, we get that

$$(\hat{x}^T A y)_{i_1} + (\hat{x}^T A y)_{i_2} = (x^T A y)_{i_1} + (x^T A y)_{i_2}.$$

Furthermore, since \hat{x} and x equal on all entries $i \in [n] \setminus \{i_1, i_2\}$, we know that

$$\sum_{i \neq i_1, i_2} (\hat{x}^T A y)_i = \sum_{i \neq i_1, i_2} (x^T A y)_i,$$

and thus

$$\begin{aligned} (\hat{x}^T A y) &= \sum_{i \neq i_1, i_2} (\hat{x}^T A y)_i + (\hat{x}^T A y)_{i_1} + (\hat{x}^T A y)_{i_2} \\ &= \sum_{i \neq i_1, i_2} (x^T A y)_i + (x^T A y)_{i_1} + (x^T A y)_{i_2} = (x^T A y). \end{aligned}$$

Therefore $\hat{x}^T A y = x^T A y$ for all column strategies y with $S(y) = J$. \square

A.3 Section 5

Proof (Lemma 8). Let (x, y) be an equilibrium, and suppose that $S(y)$ includes two equivalent strategies i and j . To prove the lemma it suffices to show that there exists an equilibrium (x, \hat{y}) with $S(\hat{y}) = S(y) \setminus \{j\}$. For this, let us take \hat{y} to be the strategy vector defined by $\hat{y}_i := y_i + y_j$, $\hat{y}_j := 0$, and $\hat{y}_x = y_x$ for all $x \in \{1, \dots, n\} \setminus \{i, j\}$. Clearly $S(\hat{y}) = S(y) \setminus \{j\}$. Thus, for each $s \in S(\hat{y})$ we have

$$(x^T B)_s \geq (x^T B)_j, \forall j \neq s,$$

since this condition holds for all $s \in S(y) \supseteq S(\hat{y})$ by Lemma 1. Furthermore, due to the definition of equivalence, a simple calculation shows that $(Ay)_s = (A\hat{y})_s$ for all $s \in \{1, \dots, k\}$. Therefore, for each $s \in S(x)$, we get again using Lemma 1 that

$$(A\hat{y})_s = (Ay)_s \geq (Ay)_j, \forall j \neq s.$$

It follows that (x, \hat{y}) satisfies both conditions of Lemma 1, and so it is indeed an equilibrium. \square