Lipschitz games, in which there is a limit \( \lambda \) (the Lipschitz value of the game) on how much a player’s payoffs may change when some other player deviates, were introduced about 10 years ago by Azrieli and Shmaya. They showed via the probabilistic method that \( n \)-player Lipschitz games with \( m \) strategies per player have pure \( \epsilon \)-approximate Nash equilibria, for \( \epsilon \geq \lambda \sqrt{8n \log(2mn)} \). Here we provide the first hardness result for the corresponding computational problem, showing that even for a simple class of Lipschitz games (Lipschitz polymatrix games), finding pure \( \epsilon \)-approximate equilibria is PPAD-complete, for suitable pairs of values \( (\epsilon(n), \lambda(n)) \). Novel features of this result include both the proof of PPAD hardness (in which we apply a population game reduction from unrestricted polymatrix games) and the proof of containment in PPAD (by derandomizing the selection of a pure equilibrium from a mixed one). In fact, our approach implies containment in PPAD for any class of Lipschitz games where payoffs from mixed-strategy profiles can be deterministically computed.

1 Introduction

The basic setting of game theory models a finite game as a finite set of players, each of whom chooses from a finite set of allowed actions (or pure strategies). Such a game maps any choice of actions by the players to payoffs for the players. It follows that if players are allowed to randomize over their actions, there is a well-defined notion of expected payoff for each player. Nash’s famous theorem [26] states that there exist randomized (or “mixed”) strategies for the players so that no player can improve their expected payoff by unilaterally deviating to play some alternative strategy. In this paper we make a standard assumption that all payoffs lie in the range \([0,1]\), and we take an interest in \( \epsilon \)-approximate Nash equilibria, in which no single player can improve their expected payoff by more than some small additive \( \epsilon > 0 \) by deviating.

In a \( \lambda \)-Lipschitz game, a deviation by any single player can change any other player’s payoffs by at most an additive \( \lambda \). This property of a multiplayer game is of course analogous to the notion of Lipschitz continuity in mathematical analysis. As shown by Azrieli and Shmaya [2], for \( \lambda \leq \epsilon / \sqrt{8n \log(2mn)} \), any \( n \)-player, \( m \)-action, \( \lambda \)-Lipschitz game has an \( \epsilon \)-approximate Nash equilibrium in pure strategies. This is shown by taking a mixed-strategy Nash equilibrium, known to exist by [26], and showing that when pure strategies are sampled from it, there is a positive probability that it will be \( \epsilon \)-approximate. As observed in [2], solutions based on mixed strategies are often criticized as being unrealistic in practice, and pure-strategy solutions are more plausible. However the existence proof of [2] is non-constructive, raising the question of how hard they are to discover.

*This author was supported by an Oxford-DeepMind Studentship during the development of this work.
Here we study the problem of computing approximate pure equilibria of Lipschitz games. In a Lipschitz polymatrix game, the effect of any one player $A$’s action on the payoffs of any other player $B$ is both bounded and independent of what the remaining players are doing. For each of $B$’s available actions, $B$’s payoff is just the sum of a collection of contributions (each at most $\lambda$) associated with the action of each other player, plus some constant shift. For these simple games, we identify values of $\lambda$ and $\epsilon$ (as functions of $n$) for which this problem is PPAD-complete, and thus unlikely to have a polynomial-time algorithm. Containment in PPAD holds for more general subclasses of Lipschitz games; in essence, any for which one can deterministically and efficiently compute payoffs associated with mixed-strategy profiles. However, for Lipschitz games whose payoff function is computed by a general circuit (or more abstractly, a payoff oracle for pure-strategy profiles), we just have containment in a randomized analogue of PPAD.

### 1.1 Background, Related Work

A key feature of the problem of computing a mixed Nash equilibrium (either an exact one, or an $\epsilon$-approximate one for some $\epsilon > 0$) is that due to the guaranteed existence of a solution that once found can be easily checked, it cannot be NP-hard unless $\text{NP} = \text{co-NP}$. Beginning with the complexity class PPAD (defined below) emerged as the means of identifying whether a class of games is likely to be hard to solve. Until recently, PPAD-completeness has mostly been used to identify computational obstacles to finding mixed equilibria rather than pure ones. In the setting of Bayes-Nash equilibrium, Filos-Ratsikas et al. obtain a PPAD-completeness result for pure-strategy solutions of a class of first-price auctions. Closer to the present paper is a PPAD-completeness result of Papadimitriou and Peng for a class of public goods games with divisible goods, where the continuous action-space of the players can represent probabilities in the solution of a corresponding 2-player game. Here instead we approximate continuous values (those taken at the gates of an arithmetic circuit for which we seek a fixed point) by the actions of a subset of the players, interpreting a given estimate as the fraction of players in such a subset who play some strategy.

### Lipschitz Games

Lipschitz games are a well-studied topic formally introduced by Azrieli and Shmaya as a natural class of games that admit approximate pure Nash equilibria. Such games arise in many common situations such as financial markets and various types of network games.

These games are assumed to have a large number of players, and introduce the (Lipschitz) restriction that every player can have at most a bounded impact on the payoffs of any other player. They exhibit many fascinating properties, such as various metrics of fault tolerance (explored in and ex-post stability of equilibria (studied in)).

Following their introduction, further work on equilibria of Lipschitz games by Daskalakis and Papadimitriou shows that for anonymous Lipschitz games, pure approximate equilibria can be computed in polynomial time via reduction to MAXFLOW, even for an approximation factor independent of the number of players (i.e., $\epsilon = O(\lambda n)$ for $\lambda$-Lipschitz, $m$-action games). When such games are restricted to 2 actions, there is an algorithm that can find such equilibria making polynomially-many approximate queries (where the query may be off by some additive error), and even a randomized algorithm running in polynomial time using only best-response queries. Exploit a concentration argument to generalize and identify pure approximate equilibria in aggregative games.

In, Goldberg and Katzman study Lipschitz games from the perspective of query complexity, in which an algorithm has black-box access to the game’s payoff function. They identify lower bounds on the number of payoff queries needed to find an equilibrium. This leaves open the problem of computing an equilibrium of a Lipschitz game that has a concise representation: if a game is known to be concisely representable, then its query complexity is low. Here we show that for concisely-representable Lipschitz games, there remains a computational obstacle. Prior to, gave a query-based algorithm for computing $1/8$-approximate Nash equilibria in $1/n$-Lipschitz games; there may be scope to improve on the constant of $1/8$, but the present paper indicates a limit to further progress.

### Polymatrix Games

Another line of work explores the computational properties of equilibria in polymatrix games introduced in and show that, for coordination-only polymatrix games (games in which each pair of players wants to agree) finding pure Nash equilibria is PLS-complete while finding mixed Nash equilibria is contained in CLS. Furthermore, shows that the problem of finding $\epsilon$-approximate Nash equilibria of general polymatrix games is PPAD-complete for some constant $\epsilon > 0$. To complement this lower bound, provide an upper bound in the form of an algorithm running in time polynomial in the number of players to find an $\epsilon$-approximate Nash equilibrium of such games for any constant $\epsilon > 0.5$ independent of the number of actions. Polymatrix games are known to be hard to solve even for sparse win-lose matrices, and for tree polymatrix games.
Here we extend the PPAD-hardness results of [29] to games having the Lipschitz property in addition to being polymatrix and binary-action. We show PPAD containment for the problem of finding approximate pure Nash equilibria of this class.

1.2 Our Contributions

In its most general form, the problem we address is, for \( \lambda \leq \epsilon / \sqrt{8n \log 4mn} \):

\[
\text{Given as input an } n\text{-player, } m\text{-action game, find either an } \epsilon\text{-approximate pure Nash equilibrium, or a witness that the game is not } \lambda\text{-Lipschitz for the above } \lambda \text{ (a witness consists of two pure-strategy profiles whose payoffs show the game is not } \lambda\text{-Lipschitz).}
\]

We will consider \( m, \epsilon, \) and \( \lambda \) to be parameters of the problem statement rather than provided in the input, however the problem is similar if these are instead passed in as input.

Notice that the two alternative kinds of solutions (an equilibrium or a witness against the \( \lambda \)-Lipschitz property) avoids a promise problem: there is no hard-to-check promise that a given game is Lipschitz. The problem statement can be taken to apply in general to any class of games whose payoff functions are efficiently computable (e.g. circuit games [30]). Efficient computability of payoffs is also a sufficient condition to guarantee that solutions are easy to check, allowing us to check the validity of either kind of solution. In the special case of Lipschitz polymatrix games of interest here, they can be presented in a straightforward format for which we can check the \( \lambda \)-Lipschitz property.

A PPAD-completeness result consists of containment in PPAD as well as PPAD-hardness. In the context of mixed-strategy Nash equilibrium computation, the following property of a class of games is sufficient for containment in PPAD:

\[
\text{Given a mixed-strategy profile, it is possible to deterministically compute the payoffs with additive error } O(\epsilon), \text{ in time polynomial in the representation size of the game, and } 1/\epsilon.
\]

This property is shared by most well-studied classes of multiplayer games, such as graphical games or polymatrix games. It does not hold for unrestricted Lipschitz games\(^2\), but it of course holds for Lipschitz polymatrix games. In this paper we obtain PPAD containment for pure-strategy approximate equilibria. The approach of [2] places the problem in “randomized PPAD”. We establish containment in PPAD via a deterministic approach to selecting a pure approximate equilibrium (Lemma 3.4). Our approach requires \( \lambda \) to be a smaller-growing function of \( \epsilon \) and \( n \) than that of [2], however there is overlap between the \((\epsilon, \lambda)\) régime where the problem is contained in (deterministic) PPAD and the one where the problem is hard for PPAD.

PPAD-hardness applies to the binary-action case where each player has two pure strategies \{1, 2\}. We reduce from the polymatrix games of [29], by considering an induced population game (details in Section 3.2), which has the Lipschitz property.

For any fixed number of actions \( m \geq 2 \), PPAD-completeness (Theorem 3.1 and Corollary 3.6) holds for carefully-chosen functions \( \epsilon(n) \) and \( \lambda(n) \). In particular, \( \lambda(n) \) needs to be sufficiently small relative to \( \epsilon(n) \) in order for our derandomization approach to achieve containment in PPAD (described in Section 3.3 Theorem 3.2). At the same time \( \lambda(n) \) needs to be large enough that we still have hardness for PPAD (Section 3.2 Theorem 3.5). Theorem 3.1 and Corollary 3.6 identify \( \epsilon(n) \), \( \lambda(n) \) for which both of these requirements hold.

2 Preliminaries

We use the following basic notation throughout.

- Boldface letters denote vectors; the symbol \( \mathbf{a} \) is used to denote a pure profile, and \( \mathbf{p} \) is used when the strategy profiles may be mixed.
- \([n]\) and \([m]\) denote the sets \{1, \ldots, n\} of players and \{1, \ldots, m\} of actions, respectively. Furthermore, \( i \in [n] \) will always refer to a player, and \( j \in [m] \) will always refer to an action.

\(^2\)For Lipschitz games presented in terms of unrestricted circuits that compute payoffs from pure-strategy profiles, one cannot deterministically and efficiently compute approximate payoffs to mixed-strategy profiles unless \( P = RP \). For games abstractly presented via a payoff query oracle, this is unconditionally not possible to achieve (regardless of any complexity class collapses).
2.1 Basic Game Theoretic and Complexity Concepts

We first introduce standard concepts of strategy profiles, payoffs, regret, and pure/mixed equilibria, followed by the complexity class PPAD. We mostly consider binary-action games (where $m = 2$) with actions labelled 1 and 2 (for more general results considering $m > 2$, see Section 4).

Types of Strategy Profile

• A pure action profile $a = (a_1, \ldots, a_n) \in [m]^n$ is an assignment of one action to each player. We use $a_{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) \in [m]^{n-1}$ to denote the set of actions played by players in $[n] \setminus \{i\}$.

• A (possibly mixed) strategy profile $p = (p_1, \ldots, p_n) \in (\Delta([m]))^n$, where $\Delta([m])$ is the probability simplex over $[m]$. When $m = 2$, let $p_i$ be the probability with which player $i$ plays action 2. Generally $p_{i,j}$ is the probability that player $i$ allocates to action $j$. The set of distributions for players in $[n] \setminus \{i\}$ is denoted $p_{-i} = (p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n)$. When $p$ contains just 0-1 values, $p$ is equivalent to some action profile $a \in [m]^n$.

• A random realization of a mixed strategy profile $p$ is a pure strategy profile $a$ such that, for every player $i$, $a_i = j$ with probability $p_{i,j}$.

Notation for Payoffs

Given player $i$, action $j$, and pure action profile $a$,

• $u_i(j, a_{-i})$ is the payoff that player $i$ obtains for playing action $j$ when all other players play the actions given in $a_{-i}$.

• $u_i(a) = u_i(a_i, a_{-i})$ is the payoff that player $i$ obtains when all players play the actions given in $a$.

• Similarly for mixed-strategy profiles: $u_i(j, p_{-i}) = \mathbb{E}_{a_{-i} \sim p_{-i}}[u_i(j, a_{-i})]$ and $u_i(p) = \mathbb{E}_{a \sim p}[u_i(a)]$.

Solution Concepts

Definition 1 (Best Response). Given a player $i$ and a strategy profile $p$, define the best response

$$\text{br}_i(p) = \arg \max_{j \in [m]} u_i(j, p_{-i}).$$

In addition, for $\epsilon > 0$, any action $j'$ satisfying

$$u_i(p) - u_i(j', p_{-i}) \leq \epsilon$$

is labelled an $\epsilon$-best response.

Definition 2 (Regret). Given a player $i$ and a strategy profile $p$, define the regret

$$\text{reg}_i(p) = \max_{j \in [m]} u_i(j, p_{-i}) - u_i(p)$$

to be the difference between the payoffs of player $i$'s best response to $p_{-i}$ and $i$'s strategy $p_i$.

Definition 3 (Equilibria). We consider the following three types of Nash equilibrium:

• An $\epsilon$-approximate Nash equilibrium ($\epsilon$-ANE) is a strategy profile $p^*$ such that, for every player $i \in [n]$, $\text{reg}_i(p^*) \leq \epsilon$.

• An $\epsilon$-well supported Nash equilibrium ($\epsilon$-WSNE) is an $\epsilon$-ANE $p^*$ for which every action $j$ in the support of $p_i^*$ is an $\epsilon$-best response to $p_{-i}$.

• An $\epsilon$-approximate pure Nash equilibrium ($\epsilon$-PNE) is a pure action profile $a$ such that, for every player $i \in [n]$, $\text{reg}_i(a) \leq \epsilon$.

The Complexity Class PPAD

We assume the reader is familiar with the classes of problems P and NP and the concepts of asymptotic behavior (for an introduction to these concepts, see e.g. [1]). Below we provide the detailed definition of PPAD, introduced in [27].

PPAD is defined by a problem that is complete for the class, the END OF LINE problem.

Definition 4 (END OF LINE). Define END OF LINE to be the problem taking as input a graph $G = (V, E)$, where the vertex set $V$ is the exponentially-large set $\{0,1\}^n$, and $E$ is encoded by two circuits $P$ and $S$, where $G$ contains directed edge $(u, v)$ if and only if $S(u) = v$ and $P(v) = u$. Every vertex has both in-degree and out-degree at most one, and $0^n \in V$ has indegree 0, outdegree 1. The problem is to output another vertex $v$ with total degree 1.
**Definition 5** (PPAD). PPAD is the set of problems many-one reducible to END OF LINE in polynomial time. A problem belonging to PPAD is PPAD-complete if END OF LINE reduces to that problem.

Here we do not use END OF LINE directly; we reduce from another pre-existing PPAD-complete problem.

### 2.2 Lipschitz Polymatrix Games

In this section we define the classes of games we consider, followed by the associated problem we solve. We will first define binary-action Lipschitz games, before extending the definition to more actions.

**Definition 6** (Lipschitz Games [2]). For any value \( \lambda \in (0, 1] \), a binary-action \( \lambda \)-Lipschitz game is a game in which a change in strategy of any given player can affect the payoffs of any other player by at most an additive \( \lambda \), i.e. for every player \( i \) and pair of action profiles \( a, a' \) with \( a_i = a_i' \),

\[
|u_i(a) - u_i(a')| \leq \lambda \|a_{-i} - a'_{-i}\|_0
\]

(equivalently, the \( \ell_1 \) norm can be used, and is more relevant when it comes to mixed strategies). Note that, under the definition of payoffs to mixed strategies presented above, because mixed strategies of binary-action games are still represented as vectors (of scalars), this can be easily extended to mixed profiles \( p, p' \) with \( p_i = p'_i \):

\[
|u_i(p) - u_i(p')| \leq \lambda \|p_{-i} - p'_{-i}\|_1.
\]

The above definition, while presented in a different way, is mathematically equivalent to that given by [2] (just provided here in a form more relevant to the current work).

In order to extend the definition above to \( m > 2 \) actions, we will require a measure of distance for probability distributions:

**Definition 7** (Total Variation Distance). The total variation distance between distributions \( D_1 \) and \( D_2 \) over discrete sample space \( X \) (each represented by \( m \)-dimensional vectors) is

\[
d_{TV}(D_1, D_2) = \frac{1}{2} \|D_1 - D_2\|_1 = \frac{1}{2} \sum_{x \in X} |\Pr_{x \sim D_1}(x) - \Pr_{x \sim D_2}(x)|.
\]

**Definition 8.** For \( m > 2 \), the pure action definition remains the same as in the \( m = 2 \) case. However, now that mixed strategies are represented by vectors of distributions, the Lipschitz property becomes

\[
|u_i(p) - u_i(p')| \leq \lambda \sum_{i'=1}^n d_{tv}(p_i, p'_{i'})
\]

where \( d_{tv} \) is the total variation distance between two distributions (i.e. an extension of the \( \ell_1 \) norm definition above).

**Remark 1.** We will rely on the total variation distance implicitly throughout the remainder of this work. It follows from Definition [8] that when a player \( i \) moves probability \( \rho \) from one action to another in a mixed strategy profile, every other player’s payoffs may be affected by at most an additive \( \lambda \rho \). In addition, because the payoff function \( u_i \) is \( \lambda \)-Lipschitz in this sense, the regret function \( \text{reg}_i \) is \( 2\lambda \)-Lipschitz in the same way, as the payoff of player \( i \)'s best response can increase with a slope of at most \( \lambda \) while the payoff of player \( i \)'s current strategy can decrease with a slope of at most \( \lambda \).

Here, we define polymatrix games, introduced in [22]:

**Definition 9** (Polymatrix Games). A polymatrix game is an \( n \)-player, \( m \)-action game in which each pair of players \( i_1, i_2 \) plays a bimatrix game and receives as payoff the sum of their \( n - 1 \) payoffs from each separate bimatrix game. If player \( i_1 \) plays action \( j_1 \) and player \( i_2 \) plays action \( j_2 \), we denote the payoff to player \( i_1 \) from this bimatrix game as \( \beta_{i_1,i_2,j_1,j_2} \).

Note that, as defined, such a game is \( \lambda \)-Lipschitz for any

\[
\lambda \geq \max_{i,i',j,j'} |\beta_{i,i',j,j'} - \beta_{i,i',j,j'}|.
\]

At most \( n^2 m^3 \) queries to the payoff function are required to ensure this inequality holds. Furthermore, if \( i \) plays \( j \), \( i \)'s payoff is a linear function of the indicator variables of the other player-strategy pairs. \( \beta_{i,i',j,j'} \) represents a small contribution, possibly negative, that \( i' \) makes to \( i \) (when \( i \) plays \( j \)) by playing \( j' \). To be valid, all total payoffs must
lie in \([0, 1]\), which can be checked from a game presented via the quantities \(\beta_{i,i',j,j'}\) by performing the following additional checks for every \(i \in [n], j \in [m]\):

\[
\sum_{i' \neq i} \max_{j' \in [m]} \beta_{i,i',j,j'} \leq 1
\]

\[
\sum_{i' \neq i} \min_{j' \in [m]} \beta_{i,i',j,j'} \geq 0
\]

There are \(2nm\) such calculations, each of which requires \(O(nm)\) operations, so an invalid game can be uncovered in polynomial time.

**Remark 2.** Lipschitz polymatrix games are a strict subset of Lipschitz games and require only \(O(n^2m^2)\) queries to learn completely and can thus be concisely represented in \(O(n^2m^2)\) space (by providing the value of each \(\beta_{i_1,i_2,j_1,j_2}\)).

We now define the main problem we consider in this work.

**Definition 10.** Define \((m, \epsilon, \lambda)\)-\(\text{PURELPG}\) to be the problem of finding \(\epsilon\)-PNEs of \(n\)-player, \(m\)-action, \(\lambda\)-Lipschitz polymatrix games, or alternatively finding a witness that the game is not \(\lambda\)-Lipschitz (note once more that \(m, \epsilon,\) and \(\lambda\) are parameters of the problem while \(n\) is provided as input). Furthermore, define \((m, \epsilon, \lambda)\)-\(\text{MIXEDLPG}\) to be the equivalent problem for mixed equilibria.

**Remark 3.** Note that, while we will often fix the number of actions \(m\), both \(\epsilon\) and \(\lambda\) are often functions of the number of players \(n\). We generally think of these as being decreasing functions of \(n\).

We are interested in the complexity of the problem, for various pairs of functions \(\epsilon(n), \lambda(n)\).

**Remark 4.** One basic observation we make is that, for any \(a \in (0, 1)\), \((m, \epsilon, \lambda)\)-\(\text{PURELPG}\) reduces to \((m, ac, a\lambda)\)-\(\text{PURELPG}\), by rescaling payoffs.

We extend the following result of [29] to pure equilibria in Lipschitz polymatrix games:

**Theorem 2.1** ([29]). There exists some constant \(\epsilon > 0\) such that given a binary-action polymatrix game, finding an \(\epsilon\)-ANE is PPAD-complete.

### 3 Results

The result we achieve is as follows:

**Theorem 3.1** (Main Result). For every constant \(m \geq 2\), there exists some constant \(\epsilon > 0\) such that, for all functions \(\lambda(n) = \Theta(n^{-3/4})\), \((m, \epsilon, \lambda)\)-\(\text{PURELPG}\) is PPAD-complete.

In fact, for any constant \(a \in (\frac{3}{4}, 1)\), Theorem 3.1 holds for \(\lambda(n) = \Theta(n^{-a})\). In order to prove Theorem 3.1, we will need to show both containment in PPAD and PPAD-hardness under these settings of the parameters.

#### 3.1 Containment in PPAD

**Theorem 3.2.** For all functions \(\epsilon(n), \lambda(n)\) satisfying

\[
\lambda(n) = \frac{1}{\text{poly}(n)}, \quad \epsilon = \lambda(n)\omega(n^{2/3})
\]

\((m, \epsilon, \lambda)\)-\(\text{PURELPG}\) \(\in\) PPAD.

In particular, we prove this when \(\epsilon(n) \geq 6\lambda(n) \sqrt{n^2m \log 3m}\). We will include the proof for \(m = 2\) (see Section 4 for the more general proof). The proof of this theorem is broken down further into two smaller steps, along the lines of [2]. First, we will exhibit sufficient settings of the parameters such that mixed approximate Nash equilibria of Lipschitz polymatrix games can be found in PPAD. Second, with the parameters set as in the statement of Theorem 3.2, we describe a derandomization technique for deriving a pure approximate equilibrium from the mixed one in polynomial time. More formally:

**Lemma 3.3.** Whenever \(\lambda(n) = \frac{1}{\text{poly}(n)}, (2, \frac{3}{8}, \lambda)\)-\(\text{MIXEDLPG}\) \(\in\) PPAD.

**Sketch.** As in the standard proof placing the problem of finding approximate Nash equilibria in PPAD (e.g. Theorem 3.1 of [10]), since \(\epsilon\) is an inverse polynomial in \(n\), the key requirement is the existence of an efficient deterministic algorithm answering mixed payoff queries. Because the payoffs to a player \(i\) in polymatrix games are linear in the actions of every other player, any mixed payoff query can be answered in time \(O(n^2m^2)\) (or simply \(O(n^2)\) when keeping the number of actions fixed). Thus \((2, \frac{3}{8}, \lambda(n))\)-\(\text{MIXEDLPG}\) \(\in\) PPAD. \(\square\)
Lemma 3.4 (Main Technical Lemma). For functions $\epsilon(n), \lambda(n)$ satisfying $\epsilon(n) \geq \lambda \sqrt[70]{n^2}$, an $\epsilon$-PNE of an $n$-player, binary-action, $\lambda$-Lipschitz polymatrix game $G$ can be derived from an $\frac{1}{8}$-ANE of $G$ in polynomial time.

Proof. Throughout this proof we implicitly use the assumption that payoffs from mixed strategy profiles can be computed in polynomial time. We will consider the (signed) discrepancy

$$d_i(p) = u_i(2, p_{-i}) - u_i(1, p_{-i})$$

($d_i$ is a $2\lambda$-Lipschitz function via the same argument as Remark 1).

Take some $\frac{1}{8}$-ANE $p^*$ of $G$. The reduction is performed in three steps:

1. Constructing from $p^*$ a well-supported Nash equilibrium $p^{(0)}$. This step is required because we will use a player’s discrepancy as a linear proxy for regret, and general approximate equilibria may allow players to allocate small probabilities to bad actions, thus decoupling the two measures (in contrast with exact equilibria, in which any mixed strategy by definition implies no discrepancy between the actions). Well-supported equilibria do not suffer from this same decoupling.

2. Iteratively converting $p^{(0)}$ into a pure action profile $a$ that is “close” to an equilibrium. To achieve this step, we will bound the rate at which the sum of all players’ discrepancies is allowed to increase, ensuring that few players experience large regret.

3. Correcting $a$ to obtain an approximate pure equilibrium $a^*$. Because of the guarantees from the previous step, we can achieve this by allowing every player with high regret to change to their best response.

Step 1: Mixed equilibrium to WSNE In this first step we construct WSNE $p^{(0)}$ from $p^*$. The idea here is to ensure that all players fall into one of the following two categories:

- Players with discrepancy that is small in absolute value
- Players playing pure strategies

In order to do this, consider any player $i$ such that $|d_i(p^*)| > \frac{\lambda}{2} \sqrt{n}$ (without loss of generality we can assume the discrepancy is negative, i.e. action 1 is significantly better). Then

$$\text{reg}_i(p^*) = -p^*_i d_i > p^*_i \frac{\lambda}{2} \sqrt{n}$$

so, since $p^*$ is an approximate equilibrium,

$$p^*_i < \frac{1}{4\sqrt{n}}.$$ 

If every such player (at most $n$) changes to playing their pure best response, then every player’s discrepancy will increase by at most $2n \lambda p^*_i$ (recall that discrepancy is $2\lambda$-Lipschitz), so in this new profile, no player playing a mixed strategy can experience discrepancy greater than

$$\frac{\lambda}{2} \sqrt{n} + 2n \lambda \frac{1}{4\sqrt{n}} = \lambda \sqrt{n}$$

while no player playing a pure strategy can experience regret greater than

$$0 + 2n \lambda \frac{1}{4\sqrt{n}} = \frac{\lambda}{2} \sqrt{n}.$$ 

Thus this new profile is a $\lambda \sqrt{n}$-WSNE, and for any player $i$ playing a mixed strategy, $d_i(p^*) \leq \lambda \sqrt{n}$. Call this $\lambda \sqrt{n}$-WSNE $p^{(0)}$. 

Step 2: WSNE to pure profile  In this step, we iteratively define intermediate strategy profiles \( \mathbf{p}^{(i)} \) and intermediate sets of players \( S^{(i)} \) who have, at any point, experienced small discrepancy (defined below to ensure that players not in \( S^{(i)} \) cannot have high regret in profile \( \mathbf{p}^{(i)} \)). Informally, for every player \( i \), \( \mathbf{p}^{(i)} \) is a strategy profile in which players \( 1, \ldots, i \) are playing pure strategies and players \( i + 1, \ldots, n \) may not be. Furthermore, the set \( S^{(i)} \) is the set of players who, in at least one of the profiles \( \mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(i)} \) have experienced discrepancy at most \( \lambda \sqrt{n} \). These will be iteratively computed to ultimately arrive at \( \mathbf{a} = \mathbf{p}^{(n)} \); a pure action profile that is in some sense “close” to an approximate pure equilibrium.

Counterintuitively, this will not always involve each player to play their best response. Instead, we will instruct each player to play the pure strategy that allows us to bound the effect on the other players. Roughly, we identify below a measure of badness \( C \) that is quadratic in the probability \( p \) assigned by a player to strategy 2 as opposed to strategy 1. In setting \( p \) to 0 or 1, we minimize the linear term in this measure, choosing 0 or 1 depending on the sign of the derivative with respect to \( p \), and the measure increases by the relatively small second-order term.

More formally, take \( \mathbf{p}^{(0)} \) as above and define

\[
S^{(0)} = \{ i' : |d_i'(\mathbf{p}^{(0)})| \leq \lambda \sqrt{n} \}
\]

(i.e. a superset of the players playing mixed strategies in \( \mathbf{p}^{(0)} \) while

\[
S^{(i)} = S^{(i-1)} \cup \{ i' : |d_i'(\mathbf{p}^{(i)})| \leq \lambda \sqrt{n} \}
\]

(adding at each step the set of players playing pure strategies in \( \mathbf{p}^{(0)} \) but experiencing small discrepancy for the first time at step \( i \); note that, due to the Lipschitz property, no player’s discrepancy can change sign without that player joining \( S^{(i)} \), so any player not in \( S^{(i)} \) must be playing their pure best response). Furthermore, define the sum of squared discrepancies cost function (which considers only the players in \( S^{(i)} \)):

\[
C(\mathbf{p}^{(i)}) = \sum_{i' \in S^{(i)}} d_i^2(\mathbf{p}^{(i)}).
\]

Note that \( C \) only considers players in the set \( S^{(i)} \), as it is meant to be a proxy for how far we are from equilibrium, and players in \( [n] \setminus S^{(i)} \) have high discrepancies but are playing their best responses and hence should not contribute to this distance to equilibrium.

It is difficult to minimize \( C \) itself, and regardless \( C \) is already only a proxy for the true objective we must minimize to achieve an approximate pure equilibrium. However, since \( d_i' \) is a multi-linear function (in fact, in the case of polymatrix games, linear), we can write

\[
d_i'(\mathbf{p}^{(i)}) = c + \ell p_i
\]

where \( c \) is some constant dependent only on \( \mathbf{p}_{-i}^{(i)} \) and \( \ell \) is a coefficient with absolute value at most \( 2\lambda \) (the Lipschitz parameter of \( d_i' \)). Squaring this, and keeping in mind that \( \mathbf{p}^{(i-1)}_{-i} = \mathbf{p}_{-i}^{(i)} \), we can expand (for every \( i' \in S^{(i-1)} \)):

\[
d_i^2(\mathbf{p}^{(i)}) - d_i^2(\mathbf{p}^{(i-1)}) = \alpha(p_i^{(i)} - p_i^{(i-1)}) + \ell^2((p_i^{(i)})^2 - (p_i^{(i-1)})^2)
\]

for some value \( \alpha_i = 2c\ell \) dependent only on \( \mathbf{p}_{-i}^{(i)} \). Taking the sum over all players \( i' \in S^{(i)} \), this yields

\[
C(\mathbf{p}^{(i)}) - C(\mathbf{p}^{(i-1)}) = A(p_i^{(i)} - p_i^{(i-1)}) + A((p_i^{(i)})^2 - (p_i^{(i-1)})^2) + K_i
\]

where

- \( A \) may be negative or non-negative.
- \( \lambda \leq 4\lambda^2 n \) (and obviously \( (p_i^{(i)})^2 - (p_i^{(i-1)})^2 \leq 1 \)).
- \( K_i \) represents the one-time contribution of at most \( \lambda^2 n \) each from any player in \( S^{(i)} \setminus S^{(i-1)} \); this discrepancy will be counted in this and all future costs. While we can’t say anything about each individual \( K_i \) (since we don’t know when exactly every player’s discrepancy will dip to this level), we note that there are at most \( n \) such contributions of \( \lambda^2 n \), i.e:

\[
\sum_{i \in [n]} K_i \leq \lambda^2 n^2
\]

(this bound is the reason we needed to start from a WSNE).
If \( p_i^{(i-1)} \in \{0, 1\} \), then set \( p_i^{(i)} = p_i^{(i-1)} \) (if a player is already playing a pure strategy, we can avoid adding to \( C \) at all on their turn). Otherwise, since we can efficiently compute \( A \) and decide whether it is negative or not, we can select \( p_i^{(i)} \) such that \( A(p_i^{(i)} - p_i^{(i-1)}) \leq 0 \) (one such option for \( p_i^{(i)} \in \{0, 1\} \) must exist). This bounds

\[
C(p^{(i)}) - C(p^{(i-1)}) \leq 4\lambda^2 n + K_i
\]

and a quick induction achieves

\[
C(p^{(n)}) \leq 4\lambda^2 n^2 + \sum_{i \in [n]} K_i \leq 4\lambda^2 n^2 + \lambda^2 n^2 = 5\lambda^2 n^2.
\]

Furthermore, \( a = p^{(n)} \) is a pure action profile.

**Step 3: Pure profile to pure equilibrium** Here we distinguish between players in \( S(n) \) and players not in \( S(n) \).

The players not in \( S(n) \) must be playing their best response in \( a \). This is because they were playing their best response in \( p^{(0)} \), and their discrepancy never approached 0 or changed sign, so their best response must never have changed. In particular, therefore, none of these players suffer any regret.

We now consider the players in \( S(n) \). Note that we have bounded the sum of the squares of the discrepancies of the players in this set to be at most \( 5\lambda^2 n^2 \). We can now also bound the number of players with regrets that are too high and fix them. To begin this final step, define \( \delta = \lambda^2 \sqrt{20n^2} \) and consider the number of players \( i \in S(n) \) such that \( d_i(a) > \delta \). This will be at most \( \frac{C(a)}{\delta^2} \). So have every player with regret at least \( \delta \) in profile \( a \) (a subset of those with high discrepancy) simultaneously switch actions. Then, once more invoking Remark 1, the maximum regret any player (in \( S(n) \) or otherwise) can experience is

\[
\delta + 2\lambda \frac{C(a)}{\delta^2} = \delta + \frac{10\lambda^3 n^2}{\delta^2} < \lambda \sqrt{70n^2}.
\]

In other words, the action profile \( a^* \) obtained after these players switch actions is a \( \lambda \sqrt{70n^2} \)-PNE. This completes the proof of Lemma 3.4 □

Finally, because a violation of the Lipschitz property can be uncovered in a quick check before commencing the above algorithm simply by ensuring all the coefficients are smaller than \( \lambda \), and [6] show that PPAD is closed under Turing reductions, Theorem 3.2 clearly follows from combining the two Lemmas above.

### 3.2 Hardness for PPAD

**The Induced Population Game**

This section introduces the approach that will be used to show PPAD-hardness. The technique involves artificially converting a general game into a Lipschitz game by treating every player \( i \) as instead a collection of many different players, each with a say in \( i \)'s ultimate strategy. This reduction was used by [2] in an alternative proof of Nash's Theorem, and by [4] to upper bound the support size of \( \epsilon \)-ANEs. More recently, [17] used a query-efficient version of this reduction to lower-bound the query complexity of computing \( \epsilon \)-PNES of general Lipschitz games.

**Definition 11.** Given a game \( G \) with payoff function \( u \), we define the population game induced by \( G \), \( G' = g_G(L) \) with payoff function \( u' \) in which every player \( i \) is replaced by a population of \( L \) players (\( v_{i\ell} \) for \( \ell \in [L] \)), each playing \( G \) against the aggregate behavior of the other \( n - 1 \) populations. More precisely, for \( \mathbf{p} \) a mixed profile of \( G' \),

\[
u'_{i\ell}(\mathbf{p}') = u_i\left(p'_{i\ell}, \mathbf{p}_{-i}\right)
\]

where

\[
p_{i\ell}' = \frac{1}{L} \sum_{\ell'=1}^L p_{i\ell'}
\]

for all \( i' \neq i \).

**Remark 5.** Note that, regardless of the Lipschitz parameter of \( G \), the induced population game \( G' = g_G(L) \) is \( \frac{1}{L} \)-Lipschitz.
Population games date back even to Nash’s thesis [25], in which he uses them to justify the consideration of mixed equilibria as a solution concept. To date, the reduction to the induced population game has been focused on proofs of existence. We show that the reduction can also be used to obtain the first PPAD-hardness result for a pure Nash equilibrium problem for a class of non-Bayesian games with discrete action spaces.

Remark 6. Note that any \( \epsilon \)-PNE of \( g_G(L) \) induces a \( \frac{1}{\epsilon} \)-uniform\(^5\) \( \epsilon \)-WSNE (and thus \( \epsilon \)-ANE) of \( G \) in which each player in \( G \) plays the aggregate behavior of their population.

**Proof of Hardness**

In Theorem 3.5 we only consider binary-action games (clearly the hardness results can be extended to games with more actions).

**Theorem 3.5.** There exists some constant \( \epsilon > 0 \) such that \( (2, \epsilon, en^{-4/5}) \)-PURELPG is PPAD-hard.

**Remark 7.** While we show this Theorem for \( \lambda = en^{-4/5} \), the same proof holds for \( \lambda = en^{-\alpha} \) for any \( \alpha \in (\frac{1}{2}, 1) \). Note that if \( \alpha \leq \frac{1}{2} \) the pure equilibrium guarantee of [2] does not hold.

**Proof of Theorem 3.5.** By Theorem 2.1 there is a constant \( \epsilon > 0 \) such that it is PPAD-hard to find \( \epsilon \)-ANEs of polymatrix games (we may assume \( \frac{1}{\epsilon} \in \mathbb{N} \)). We reduce the problem of computing an \( \epsilon \)-ANE of polymatrix game \( G \) to finding an \( \epsilon \)-PNE of Lipschitz polymatrix game \( G' \) as follows. Consider the induced population game \( G' = g_G(L) \) for \( L = n^{4/5} \). This game has \( N = (n/\epsilon)^5 \) players and is \( \epsilon N^{-4/5} \)-Lipschitz. Thus (for large enough \( n \)) it is guaranteed to have an \( \epsilon \)-PNE. Furthermore, \( G' \) is still a polymatrix game, as the payoff to any player \( v_i \) is simply the sum of the payoffs from \( N - L \) games played with the players \( v_i' \) for \( i' \neq i \).

Now, because the payoffs of any pure action profile of \( G' \) can be derived from payoffs of mixed strategy profiles of \( G \) (which can be computed in polynomial time), this entire reduction occurs in polynomial time. Thus there is a constant \( \epsilon > 0 \) such that, for \( \lambda(n) = en^{-4/5} \), \( (2, \epsilon, \lambda) \)-PURELPG is PPAD-hard. \( \square \)

Combining Theorems 3.2 and 3.5 yields Theorem 3.1. In fact, by scaling the payoffs in Theorem 3.1 as described in Remark 3, we obtain:

**Corollary 3.6.** Fix some constant \( \alpha \in (\frac{2}{3}, 1) \). For any constant \( m \geq 2 \), non-increasing function \( \epsilon(n) = \frac{1}{\text{poly}(n)} \), and \( \lambda(n) = \Theta(\epsilon n^{-\alpha}) \), \((m, \epsilon, \lambda)\)-PURELPG is PPAD-complete.

### 4 Upper Bound for Additional Actions

**Theorem 4.1** (Restatement of Theorem 3.2). For functions \( \epsilon(n), \lambda(n) \) satisfying

\[
\lambda(n) = \frac{1}{\text{poly}(n)}, \quad \epsilon(n) \geq 6\lambda(n)^{3/2}n^2m \log 3m
\]

\( (m, \epsilon, \lambda) \)-PURELPG is PPAD.

Once again, the proof of this theorem is broken down further into two smaller steps:

**Lemma 4.2** (Extension of Lemma 3.3). For any fixed \( m \geq 2 \), \( \lambda(n) = \frac{1}{\text{poly}(n)} \), (\( m, \left(\frac{m-1}{m}\right)^2 \lambda, \lambda \))-MIXEDLPG is PPAD.

The proof of Lemma 4.2 is essentially the same as that of Lemma 3.3.

**Lemma 4.3** (Extension of Lemma 3.4). For functions \( \epsilon(n), \lambda(n) \) satisfying

\[
\epsilon(n) \geq 6\lambda(n)^{3/2}n^2m \log 3m
\]

an \( \epsilon(n) \)-PNE of an \( n \)-player, \( m \)-action, \( \lambda \)-Lipschitz polymatrix game \( G \) can be derived from an \( \left(\frac{m-1}{m}\right)^2 \lambda \)-ANE of \( G \) in polynomial time.

\(^5\)a \( k \)-uniform mixed strategy is one in which each action of each player is assigned a probability that is a discrete multiple of \( k \)
Proof. We will use the following values in this proof. Let
\[ \epsilon_0 = \left( \frac{m-1}{m} \right)^2 \lambda \]
\[ \epsilon_1 = 2\sqrt{2n\lambda\epsilon_0} \]
Then, in order to prove Lemma 4.3 we need to convert an \( \epsilon_0 \)-ANE to an \( \epsilon \)-PNE. We will do so in the same three steps as in the proof of Lemma 3.4 but more care must be taken when generalizing the proof.

The primary generalization required is the transition from considering discrepancy to considering variance. Whereas in Lemma 3.4 there was a well-defined notion of discrepancy that, when squared, could be expressed as quadratic in the probabilities assigned to the actions of a given player, such a concept does not easily translate to games with more actions. In order to rely on similar techniques, we will instead consider the variance of a subset of each player’s actions. The exact subset to be considered will be described below, and throughout this proof will be referred to as the ‘relevant’ set, as it will contain the set of actions that may become relevant in our analysis of the algorithm.

We will proceed along the same lines as the proof of Lemma 3.4 with the modified approach.

Step 1: Mixed equilibrium to WSNE. We begin with an \( \epsilon_0 \)-ANE \( p^* \). Consider any player \( i \) and assume that player \( i \) puts probability \( p \) on actions with regret greater than
\[ \delta_0 = \sqrt{2(n-1)\lambda\epsilon_0} \]
The regret of this player is at least \( p\delta_0 \), so it must be the case that \( p \leq \epsilon_0 / \delta_0 \).
This being the case, if every player simultaneously moves the probability from such actions to their best response, then any given player’s worst-case regret becomes
\[ \delta_0 + 2(n-1)\lambda p \leq \delta_0 + \frac{2(n-1)\lambda\epsilon_0}{\delta_0} \leq 2\sqrt{2n\lambda\epsilon_0} = \epsilon_1. \]
Thus an \( \epsilon_0 \)-ANE can be converted easily to an \( \epsilon_1 \)-WSNE.

Step 2: WSNE to pure profile. This is, once again, the most involved step. Unlike Step 1 (which occurred simultaneously) and Step 3 (which will do the same), this step will take place for each player in turn. We will iteratively define intermediate strategy profiles \( p^{(i')} \) in which players \( 1, \ldots, i' \) are playing pure strategies and players \( i' + 1, \ldots, n \) may not be. Ultimately, we will arrive at \( a = p^{(n)} \): a pure action profile that is in some sense ‘close’ to an approximate pure equilibrium.

Counterintuitively, this will not always involve instructing each player to play their best response when their turn arises. Instead, we will instruct each player to play the pure strategy that allows us to bound its effect on the other players. Roughly, we identify a measure of badness that is quadratic in the probability assigned by a player to each action. In selecting a pure action, we minimize the linear term in this measure depending on the direction of the gradient, and the measure increases by the relatively small second-order term. The quadratic measure we select will be the sum of the variances of subsets of the players’ actions.

More specifically, we define the following concepts:

| Notation          | Definition                                                                 |
|-------------------|---------------------------------------------------------------------------|
| \( S_{i'} \)      | Player \( i \)'s relevant set of actions                                   |
| \( m_{i'} \)      | \( |S_{i'}| \), i.e. the number of relevant actions player \( i \) has after step \( i' \) |
| \( u_i(S, t) \)   | the length-\( |S| \) restriction of player \( i \)'s payoff vector after step \( t \) to the actions in set \( S \) |
| \( \mu_{i}(S, t) \) | the average value of \( u_i(S, t) \), i.e. the payoff to player \( i \) after step \( t \) for playing the uniform mixed strategy over all actions in player set \( S \) |
| \( \sigma_i^2(S, t) \) | the variance of the values in \( u_i(S, t) \)                                |
| \( \text{reg}_i(j, t) \) | the regret player \( i \) experiences for playing action \( j \) after step \( t \) |
Now, we will define the relevant set via the following recurrence relation. The base case will be:

\[ S_i^{(0)} = \{ j \mid \text{reg}_i(j, 0) \leq \epsilon_1 \}. \]

The recurrence is:

\[
S_i^{(i' + 1)} = S_i^{(i')}
\]

\[
\mu = \mu_i(S_i^{(i' + 1)}, i' + 1)
\]

\[
\text{while } (\max_{j \notin S_i^{(i' + 1)}} u_i(j, i' + 1) \geq \mu) :
\]

\[
S_i^{(i' + 1)} = S_i^{(i' + 1)} \cup \{ \arg\max_{j \notin S_i^{(i' + 1)}} u_i(j, i' + 1) \}
\]

\[
\mu = \mu_i(S_i^{(i' + 1)}, i' + 1)
\]

In other words, \( S_i^{(i' + 1)} \) is the superset of \( S_i^{(i')} \) obtained by repeatedly adding player \( i \)'s highest-paying action to \( S_i^{(i' + 1)} \) and recalculating the value of \( \mu_i(S_i^{(i' + 1)}, i' + 1) \) until doing so would decrease its value. We call \( S_i^{(i')} \) the relevant set because it consists of the actions that, at any point during the algorithm’s run, are worth consideration in our attempt to bound the total regret experienced by all players.

**Step 2 of the algorithm will thus proceed as follows.** During step \( i' \), player \( i' \) will first change their strategy to the pure strategy that has the least impact on the linear term of the sum of the variances of all other players’ relevant sets. Then every player will recalculate their relevant sets using the new payoffs considering the changed strategy of player \( i' \).

There are three values we need to bound here:

1. The sum of the variances of \( S_i^{(0)} \), the players’ initial relevant sets:

\[
\sum_{i=1}^{n} \sigma_i^2\left( S_i^{(0)}, 0 \right).
\]

2. How much larger the sum of the variances of \( S_i^{(i' + 1)} \) can be than the sum of the variances of \( S_i^{(i')} \) (between steps), adding to a total over all \( n \) steps of:

\[
\sum_{i=1}^{n} \sum_{i' = 1}^{n} \sigma_i^2\left( S_i^{(i')}, i' \right) - \sigma_i^2\left( S_i^{(i' - 1)}, i' \right).
\]

3. How much the sum of the variances of \( S_i^{(i')} \), can increase during step \( i' \), adding to a total over all \( n \) steps of:

\[
\sum_{i=1}^{n} \sum_{i' = 1}^{n} \sigma_i^2\left( S_i^{(i' - 1)}, i' \right) - \sigma_i^2\left( S_i^{(i' - 1)}, i' - 1 \right).
\]

Note that the sum of these three values is:

\[
\sum_{i=1}^{n} \sigma_i^2\left( S_i^{(n)}, n \right)
\]

i.e. the sum of the variances of the relevant sets at the end of the entire Step 2.

We begin with quantity (1). The variance of any set of points within the interval \([a, b]\) is at most \((b - a)^2/4\) (see, for example, [3]), so quantity (1) is at most:

\[
n \frac{\epsilon_1^2}{4} = 2n^2 \lambda \epsilon_0 = 2 \left( n \lambda (m - 1) \right)^2.
\]

Next, we bound quantity (2). Quantity (2) considers no change in the strategy profile - all payoffs considered are based on strategy \( p^{(i')} \). Then, for every player, actions are added to the set \( S_i^{(i')} \) one by one (each time increasing the average value of the payoffs) to form the set \( S_i^{(i' + 1)} \). The critical observation is that, in addition to providing a lower bound (the current average) on the payoff of this newly added action, we can also ensure an upper bound of \( \mu_i\left(S_i^{(i')}, i' \right) + 2\lambda \), as the action must not have been relevant in the previous step. So specifically, consider a set of
k points \( \{x_1, \ldots, x_k\} \) with mean \( \mu \) and variance \( \sigma^2 \) and consider adding a point \( x_{k+1} \) that is within an additive \( 2\lambda \) of \( \mu \). We calculate the change in variance below, keeping in mind that if we subtract \( \mu \) from every point the variance remains the same:

\[
\text{Var}(x_1, \ldots, x_{k+1}) - \sigma^2 = \text{Var}(x_1 - \mu, \ldots, x_{k+1} - \mu) - \sigma^2
\]

\[
= \left( \frac{1}{k+1} \sum_{j=1}^{k+1} (x_j - \mu)^2 \right) - \left( \frac{1}{k+1} \left( \sum_{j=1}^{k} (x_j - \mu) \right)^2 \right) - \sigma^2
\]

\[
= \left( \frac{1}{k+1} \left( (x_{k+1} - \mu)^2 + \sum_{j=1}^{k} (x_j - \mu)^2 \right) \right) - \left( \frac{1}{k+1} \left( (x_{k+1} - \mu) + \sum_{j=1}^{k} (x_j - \mu) \right)^2 \right) - \sigma^2
\]

\[
= \frac{(x_{k+1} - \mu)^2}{k+1} + \frac{k}{k+1} \sigma^2 - \frac{(x_{k+1} - \mu)^2}{(k+1)^2} - \sigma^2
\]

\[
= \frac{1}{k+1} \left( \frac{k}{k+1}(x_{k+1} - \mu)^2 - \sigma^2 \right)
\]

where we can go from step 4 to step 5 using the fact that

\[
\sum_{j=1}^{k} (x_j - \mu) = 0.
\]

So, since \( |x_{k+1} - \mu| \leq 2\lambda \), the largest possible increase in variance is

\[
\frac{4k\lambda^2}{(k+1)^2}
\]

(and, in fact, adding a point will usually decrease the variance, but there’s no need to get that specific because this bounded increase will serve our purpose for constant values of \( m \)). If we add this contribution for the maximum possible \( m - 1 \) actions (the relevant set will never start empty) and all \( n \) players, we see

\[
\sum_{i=1}^{n} \sum_{k=1}^{m-1} \frac{4k\lambda^2}{(k+1)^2} \leq n \sum_{k=1}^{m-1} \frac{4\lambda^2}{k} = 4n\lambda^2H_{m-1} < 4n\lambda^2(\log(m-1) + 1).
\]

Finally, we need to bound quantity (3). Note that, given \( u_i \left( S_i^{(i'-1)}, i' - 1 \right) \), the quantity \( u_i \left( S_i^{(i'-1)}, i' \right) \) and its mean and variance are all functions of the strategy vector \( \mathbf{p}_{i'} \) of player \( i' \). In particular, we can write:

\[
u_i(S_i^{(i'-1)}, i') - 1\mu_i(S_i^{(i'-1)}, i') = \mathbf{c}_{i'}^{(i)} + \mathbf{L}_{i'}^{(i)} \mathbf{p}_{i'}^{(i')}
\]

for some constant \( m_{i'}^{(i'-1)} \times 1 \) vector \( \mathbf{c}_{i'}^{(i')} \) (which may depend on the known values \( u_i \left( S_i^{(i'-1)}, i' - 1 \right) \) and \( m_{i'}^{(i'-1)} \times m \) coefficient matrix \( \mathbf{L}_{i'}^{(i')} \) in which each element has magnitude no more than \( m_{i'}^{m-1} \lambda \). Propagating this expression:

\[
\sigma_i^2(S_i^{(i'-1)}, i') = \frac{1}{m_{i'}^{(i'-1)}} \left\| u_i(S_i^{(i'-1)}, i') - 1\mu_i(S_i^{(i'-1)}, i') \right\|^2 \leq \frac{1}{m_{i'}^{(i'-1)}} \left( \left\| \mathbf{c}_{i'}^{(i')} \right\|^2 + \left\| \mathbf{L}_{i'}^{(i')} \mathbf{p}_{i'}^{(i')} \right\|^2 + 2\mathbf{c}_{i'}^{(i')} \cdot \mathbf{L}_{i'}^{(i')} \mathbf{p}_{i'}^{(i')} \right). \quad (1)
\]

If we specifically consider the mixed term:

\[
2\mathbf{c}_{i'}^{(i')} \cdot \mathbf{L}_{i'}^{(i')} \mathbf{p}_{i'}^{(i')} = 2\mathbf{c}_{i'}^{(i')} \tau \left( \mathbf{L}_{i'}^{(i')} \mathbf{p}_{i'}^{(i')} \right) = 2 \left( \mathbf{c}_{i'}^{(i')} \tau \mathbf{L}_{i'}^{(i')} \right) \mathbf{p}_{i'}^{(i')} = \mathbf{b}_{i'}^{(i')} \mathbf{p}_{i'}^{(i')} \quad (2)
\]
for some vector $b_i^{(t')} \in \mathbb{R}^m$. So now we can consider the quantity we’re really after. We focus on (for some $i' \in [n]$) the sum over all players $i \in [n]$ of the changes in $\sigma_i^2 \left(S_i^{(t'-1)}, \cdot \right)$ after step $i'$:

$$
\sum_{i=1}^{n} \sigma_i^2 \left(S_i^{(t'-1)}, i' \right) - \sigma_i^2 \left(S_i^{(t'-1)}, i' - 1 \right).
$$

(3)

We want to minimize this value as best we can, as our goal is to end up with small variance among the set of relevant actions. While the variance is not a very good proxy for the regret (see Equation 5 below), it is a value that we will be able to successfully bound. So, substituting Equations 1 and 2 into Equation 3, we see that the change in this value across step $i'$ is

$$
\sum_{i=1}^{n} \frac{1}{m_i^{(t'-1)}} \left( \left\| c_i^{(t')} \right\|^2 + \left\| L_i \left(p_i^{(t')} \right) p_i^{(t')} \right\|^2 + b_i^{(t')}^T \left(p_i^{(t')} - p_i^{(t'-1)} \right) \right)
$$

$$
= \sum_{i=1}^{n} \frac{1}{m_i^{(t'-1)}} \left( \left\| L_i \left(p_i^{(t')} \right) p_i^{(t')} \right\|^2 + b_i^{(t')}^T \left(p_i^{(t')} - p_i^{(t'-1)} \right) \right)
$$

$$
\leq \sum_{i=1}^{n} \frac{1}{m_i^{(t'-1)}} \left( \frac{m-1}{m} \lambda \right)^2 + b_i^{(t')}^T \left(p_i^{(t')} - p_i^{(t'-1)} \right)
$$

$$
= b^{(t')}^T \left(p_i^{(t')} - p_i^{(t'-1)} \right) + \sum_{i=1}^{n} \frac{1}{m_i^{(t'-1)}} \left( \frac{m-1}{m} \lambda \right)^2
$$

(4)

for some vector $b^{(t')} \in \mathbb{R}^m$ (the sum of the vectors $\frac{1}{m_i^{(t'-1)}}b_i^{(t')}$. So, ultimately, our algorithm aims to minimize this upper bound by ensuring that the first term is not positive. Recall that $p_i^{(t'-1)}$ is a (given) vector of non-negative real numbers with $\left\| p_i^{(t'-1)} \right\|_1 = 1$, while $p_i^{(t')}$, is a (desired) vector containing $m - 1$ zeros and a single one. Consider the element $b (at index j)$ of $b^{(t')}$ with the smallest value among those indices in the relevant set $S_j^{(t')}$ (note that player $i'$s entire support will be contained in $S_j^{(t')}$). Set $b = e_j$. This ensures that the first term in Equation 4 is not positive. So, proceeding iteratively as described, this ensures

$$
\sum_{i=1}^{n} \sum_{i'=1}^{n} \sigma_i^2 \left(S_i^{(t'-1)}, i' \right) - \sigma_i^2 \left(S_i^{(t'-1)}, i' - 1 \right) \leq \sum_{i=1}^{n} \sum_{i'=1}^{n} \frac{1}{m_i^{(t'-1)}} \left( \frac{m-1}{m} \lambda \right)^2 \leq \left( \frac{m-1}{m} n \lambda \right)^2
$$

We can now bound:

- The maximum initial sum of variances of all players at $2 \left( \frac{n \lambda (m-1)}{m} \right)^2$
- The maximum contribution to the sum of variances of all players from adding actions to the relevant set at $4n \lambda^2 (\log (m - 1) + 1)$
- The maximum contribution to the sum of variances of all players from purifying the equilibrium at $\left( \frac{m-1}{m} n \lambda \right)^2$

Adding these together, the maximum sum of variances of actions in the relevant sets of players in $p^{(n)}$ is

$$
\left( \frac{m-1}{m} n \lambda \right)^2 + 4n \lambda^2 (\log (m - 1) + 1) + 2 \left( \frac{n \lambda (m-1)}{m} \right)^2
$$

$$
= 3 \left( \frac{m-1}{m} n \lambda \right)^2 + 4n \lambda^2 (\log (m - 1) + 1))
$$

$$
< 8n^2 \lambda^2 \log 3m
$$

for $m > 2$. 

14
Step 3: Pure profile to pure equilibrium

Finally, for any value of $\delta_1$, if any player has regret at least $\delta_1$, the variance of their relevant actions must be at least

$$\frac{\delta_1^2}{2m} \quad (5)$$

(there must be at least a pair of relevant actions that are a distance of $\delta_1$ apart to achieve this regret, so the lowest-variance option is realized when all remaining actions yield regret $\delta_1/2$). So allow every player with regret greater than $\delta_1$ to instead change to playing their best response (there will be at most

$$\frac{16n^2 \lambda^2 m \log 3m}{\delta_1^2}$$

such players). As each of these players can add at most $2\lambda$ to the regret of any other player (the payoff of the best response could increase by $\lambda$ while that of the chosen action could decrease by $\lambda$), the maximum regret any player can experience is

$$\delta_1 + 2\lambda \frac{16n^2 \lambda^2 m \log 3m}{\delta_1^2}.$$ Optimizing for $\delta_1$, this value is at most

$$6\lambda \sqrt{\frac{n^2 m \log 3m}{\delta_1}}.$$$

5 Discussion

Our PPAD-hardness result is rather strongly negative, since Lipschitz polymatrix games are a quite a restricted subset of either Lipschitz or polymatrix games in general. There may be scope to broaden the $(\epsilon, \lambda)$ values for which Lipschitz games are hard to solve. On the other hand, there are more positive results for computing approximate Nash equilibria of Lipschitz games, with further scope for progress.

In particular, Theorem 3.2 is unable to place the problem of finding $\epsilon$-PNEs of $\lambda$-Lipschitz polymatrix games in PPAD for the total range of values guaranteed by [2] (while they guarantee existence for $\epsilon \geq \lambda \Omega(\sqrt{n \log n})$, the above successfully computes equilibria when $\epsilon \geq \lambda \Omega(n^{2/3})$). It would be of future interest to determine if this is a characteristic of our choice of algorithm/analysis, or an indication of a barrier to the complete derandomization of computing these equilibria.

Furthermore, it may even be the case that Lipschitz polymatrix games, like anonymous Lipschitz games ([12]), guarantee pure equilibria for a smaller value of $\epsilon$. It would be interesting to determine how close to a complete derandomization we are able to achieve. There are obvious obstacles to finding a deterministic lower bound asymptotically stronger than that of [2], as such a discovery would resolve the P vs. BPP question.

References

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.
[2] Yaron Azrieli and Eran Shmaya. Lipschitz games. *Math. Oper. Res.*, 38(2):350–357, 2013.
[3] Yakov Babichenko. Best-reply dynamics in large binary-choice anonymous games. *Games Econ. Behav.*, 81:130–144, 2013.
[4] Yakov Babichenko. Small support equilibria in large games. *CoRR*, abs/1305.2432, 2013.
[5] Dimitri P Bertsekas and John N Tsitsiklis. *Introduction to Probability Vol. 1*. Athena Scientific, 2002.
[6] Samuel R. Buss and Alan S. Johnson. Propositional proofs and reductions between NP search problems. *Ann. Pure Appl. Log.*, 163(9):1163–1182, 2012.
[7] Yang Cai and Constantinos Daskalakis. On minmax theorems for multiplayer games. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 217–234. SIAM, 2011.
[8] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *J.ACM*, 56(3):14:1–14:57, 2009.
[9] Rachel Cummings, Michael J. Kearns, Aaron Roth, and Zhiwei Steven Wu. Privacy and truthful equilibrium selection for aggregative games. In Web and Internet Economics - 11th International Conference, WINE, volume 9470 of LNCS, pages 286–299. Springer, 2015.

[10] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. Commun. ACM, 52(2):89–97, 2009.

[11] Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In Dana Randall, editor, Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 790–804. SIAM, 2011.

[12] Constantinos Daskalakis and Christos H. Papadimitriou. Approximate Nash equilibria in anonymous games. J. Econ. Theory, 156:207–245, 2015.

[13] Joyee Deb and Ehud Kalai. Stability in large bayesian games with heterogeneous players. J. Econ. Theory, 157:1041–1055, 2015.

[14] Argyrios Deligkas, John Fearnley, and Rahul Savani. Tree polymatrix games are ppad-hard. In 47th International Colloquium on Automata, Languages, and Programming, ICALP, volume 168 of LIPIcs, pages 38:1–38:14, 2020.

[15] Argyrios Deligkas, John Fearnley, Rahul Savani, and Paul G. Spirakis. Computing approximate Nash equilibria in polymatrix games. Algorithmica, 77(2):487–514, 2017.

[16] Aris Filos-Ratsikas, Yiannis Giannakopoulos, Alexandros Hollender, Philip Lazos, and Diogo Poças. On the complexity of equilibrium computation in first-price auctions. In EC ’21: The 22nd ACM Conference on Economics and Computation, pages 454–476. ACM, 2021.

[17] Paul W. Goldberg and Matthew J. Katzman. Lower bounds for the query complexity of equilibria in Lipschitz games. In Algorithmic Game Theory - 14th International Symposium, SAGT, volume 12885 of Lecture Notes in Computer Science, pages 124–139. Springer, 2021.

[18] Paul W. Goldberg, Francisco J. Marmolejo Cossío, and Zhiwei Steven Wu. Logarithmic query complexity for approximate Nash computation in large games. Theory Comput. Syst., 63(1):26–53, 2019.

[19] Paul W. Goldberg and Aaron Roth. Bounds for the query complexity of approximate equilibria. ACM Trans. Economics and Comput., 4(4):24:1–24:25, 2016.

[20] Paul W. Goldberg and Stefano Turchetta. Query complexity of approximate equilibria in anonymous games. J. Comput. Syst. Sci., 90:80–98, 2017.

[21] Ronen Gradwohl and Omer Reingold. Fault tolerance in large games. Games Econ. Behav., 86:438–457, 2014.

[22] E. Janovskaja. Equilibrium points in polymatrix games. Lithuanian Mathematical Journal, 8:381–384, April 1968.

[23] Zhengyang Liu, Jiawei Li, and Xiaotie Deng. On the approximation of Nash equilibria in sparse win-lose multiplayer games. In Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI, pages 5557–5565. AAAI Press, 2021.

[24] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. Theor. Comput. Sci., 81(2):317–324, 1991.

[25] John Nash. Non-Cooperative Games. PhD thesis, Princeton University, May 1950.

[26] John Nash. Non-cooperative games. Annals of mathematics, pages 286–295, 1951.

[27] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. J. Comput. Syst. Sci., 48(3):498–532, 1994.

[28] Christos H. Papadimitriou and Binghui Peng. Public goods games in directed networks. In EC ’21: The 22nd ACM Conference on Economics and Computation, pages 745–762. ACM, 2021.

[29] Aviad Rubinstein. Inapproximability of Nash equilibrium. SIAM J. Comput., 47(3):917–959, 2018.

[30] Grant Schoenebeck and Salil P. Vadhan. The computational complexity of Nash equilibria in concisely represented games. ACM Trans. Comput. Theory, 4(2):4:1–4:50, 2012.