Settling the complexity of computing approximate two-player Nash equilibria

Aviad Rubinstein
UC Berkeley
aviad@eecs.berkeley.edu

Abstract—We prove that there exists a constant \(\epsilon > 0\) such that, assuming the Exponential Time Hypothesis for \(\text{PPAD}\), computing an \(\epsilon\)-approximate Nash equilibrium in a two-player \(n \times n\) game requires time \(n^{\log^{1-\epsilon} n}\). This matches (up to the \(o(1)\) term) the algorithm of Lipton, Markakis, and Mehta [54].

Our proof relies on a variety of techniques from the study of probabilistically checkable proofs (PCP); this is the first time that such ideas are used for a reduction between problems inside \(\text{PPAD}\).

En route, we also prove new hardness results for computing Nash equilibria in games with many players. In particular, we show that computing an \(\epsilon\)-approximate Nash equilibrium in a game with \(n\) players requires \(2^{\tilde{O}(n)}\) oracle queries to the payoff tensors. This resolves an open problem posed by Hart and Nisan [43], Babichenko [13], and Chen et al. [28]. In fact, our results for \(n\)-player games are stronger: they hold with respect to the \((\epsilon, \delta)\)-WeakNash relaxation recently introduced by Babichenko et al. [15].

Keywords—Computational complexity

I. INTRODUCTION

For the past decade, the central open problem in equilibrium computation has been whether two-player Nash equilibrium admits a PTAS. We had good reasons to be hopeful: there was a series of improved approximation ratios [51], [34], [33], [23], [67] and several approximation schemes for special cases [48], [35], [3], [17]. Yet most interesting are two inefficient algorithms for two-player Nash:

- the classic Lemke-Howson algorithm [53] finds an exact Nash equilibrium in exponential time; and
- a simple algorithm by Lipton, Markakis, and Mehta [54] finds an \(\epsilon\)-Approximate Nash Equilibrium in time \(n^{\tilde{O}(\log n)}\).

Although the Lemke-Howson algorithm takes exponential time, it has a special structure which places the problem inside the complexity class \(\text{PPAD}\) [57]; i.e. it has a polynomial time reduction to the canonical problem \(\text{EndoFALINE}\):

Definition 1.1 (\textit{EndoFALINE} [32]). Given two circuits \(S\) and \(P\), with \(m\) input bits and \(m\) output bits each, such that \(P(0^m) = 0^m \neq S(0^m)\), find an input \(x \in \{0, 1\}^m\) such that \(P(S(x)) \neq x\) or \(S(P(x)) \neq x \neq 0^m\).

Proving hardness for problems in \(\text{PPAD}\) is notoriously challenging because they are total, i.e. they always have a solution, so the standard techniques from \(\text{NP}\)-hardness do not apply. By now, however, we know that exponential and polynomial approximations for two-player Nash are \(\text{PPAD}\)-complete [32], [29], and so is \(\epsilon\)-approximation for games with \(n\) players [62].

However, \(\epsilon\)-approximation for two-player Nash is unlikely to have the same fate: otherwise, the quasi-polynomial algorithm of [54] would refute the Exponential Time Hypothesis for \(\text{PPAD}\):

Hypothesis 1 (ETH for \(\text{PPAD}\) [15]). Solving \(\text{EndoFALINE}\) requires time \(2^{\tilde{O}(n)}\).\(^2\)

Thus the strongest hardness result we can hope to prove (given our current understanding of complexity\(^2\)) is a quasi-polynomial hardness that sits inside \(\text{PPAD}\):

Theorem 1.2 (Main Theorem). There exists a constant \(\epsilon > 0\) such that, assuming ETH for \(\text{PPAD}\), finding an \(\epsilon\)-Approximate Nash Equilibrium in a two-player \(n \times n\) game requires time \(T(n) = n^{\log^{1-\epsilon} n}\).

A. TECHNIQUES

Given an \(\text{EndoFALINE}\) instance of size \(n\), we construct a two-player \(N \times N\) game for \(N = 2^n^{1/2+o(1)}\) whose approximate equilibria correspond to solutions to the \(\text{EndoFALINE}\) instance. Thus, assuming the “ETH for \(\text{PPAD}\)”, finding an approximate equilibrium requires time \(2^n = N^{\log^{1-\epsilon} N}\).

The main steps of the final construction are: (i) reducing \(\text{EndoFALINE}\) to a new discrete problem which we call \(\text{LocaLEndoFALINE}\); (ii) reducing \(\text{LocaLEndoFALINE}\) to a problem of finding an approximate Brouwer fixed point; (iii) reducing from Brouwer fixed point to finding an \(\tilde{O}(n)\) Nash equilibrium in a multiplayer game over \(n^{1/2+o(1)}\) players with \(n^{o(1)}\) actions each; and (iv) reducing to the two-player game.

The main novelty in the reduction is the use of techniques such as error correcting codes and probabilistically checkable proofs (PCPs) inside \(\text{PPAD}\). In particular, the way we use PCPs in our proof is very unusual.

1In the literature the problem has been called \(\text{EndoFTheLINE}\); we believe that the name \(\text{EndoFALINE}\) is a more accurate description.

2As usual, \(n\) is the size of the description of the instance, i.e. the size of the circuits \(S\) and \(P\).

3Given our current understanding of complexity, refuting ETH for \(\text{PPAD}\) seems unlikely: there are matching black-box lower bounds [45], [19]. Recall that the \(\text{NP}\)-analoague ETH [47] is widely used (e.g. [49], [55], [1], [25], [30]), often in stronger variants such as SETH [46], [26] and \(\text{NSETH}\) [27].
Constructing the first gap: showing hardness of \eps-
SuccinctBrouwer_2

The first step in all known PPAD-hardness results for (ap-
proximate) Nash equilibrium is reducing ENDofALINE to
the problem of finding an (approximate) Brouwer fixed point
of a continuous, Lipschitz function \( f : [0, 1]^n \to [0, 1]^n \). Let
\( \epsilon > 0 \) be an arbitrarily small constant. Previously, the state
of the art for computational hardness of approximation of
Brouwer fixed points was:

Theorem I.3 ([62], informal). It is PPAD-hard to find an
\( x \in [0, 1]^n \) such that \( \| f (x) - x \|_\infty \leq \epsilon \).

Here and for the rest of the paper, all distances are relative;
in particular, for \( x \in [0, 1]^n \) and \( p < q \), we have \( \| x \|_p \leq \| x \|_q \).

Theorem I.3 implied that it is hard to find an \( x \) such that
\( f (x) \) is approximately equal to \( x \) on every coordinate. The
first step in our proof is to strengthen this result to obtain
hardness of approximation with respect to 2-norm:

Theorem I.4 (Informal). It is PPAD-hard to find an \( x \in
[0, 1]^n \) such that \( \| f (x) - x \|_2 \leq \epsilon \).

Now, even finding an \( x \) such that \( f (x) \) is approximately

equal to \( x \) on most of the coordinates is already PPAD-hard.

Theorem I.3 was obtained by adapting a construction due
to Hirsch, Papadimitriou, and Vavasis [45]. The main idea
is to partition the \([0, 1]^n\) into \( 2^n \) subcubes, and consider
the grid formed by the subcube-centers; then embed a
path (in fact, many paths and cycles when reducing from
ENDofALINE) along an arbitrary sequence of neighboring
grid-points/subcube-centers. The function is carefully
defined along the embedded path, guaranteeing both Lipschitz
continuity and that approximate endpoints occur only near
subcube-centers corresponding to ends of paths.

Here we observe that if we want a larger displacement
(in particular, constant relative 2-norm) we actually want
the consecutive vertices on the path to be as far as possible
from each other. We thus replace the neighboring grid-points
with their encoding by an error correcting code.

The first obstacle to using PCP-like techniques for prob-
lems in PPAD is their totality (i.e. a solution always exists).
For NP-hard problems, the PCP verifier expects the proof
to be encoded in some error correcting code. If the proof
is far from any codeword, the verifier detects that (with
high probability), and immediately rejects. For problems in
PPAD (more generally, in TFNP) this is always tricky
because it is not clear what does it mean "to reject". Hirsch
et al.'s construction has the following useful property: for
the vast majority of \( x \)'s (in particular, all \( x \)'s far from
the embedding of the paths) the displacement \( f (x) - x \) is
the same default displacement. Thus, when an \( x \) is too far from
any codeword to faithfully decode it, we can simply apply
the default displacement.

We note that Theorem I.4 is already significant enough to
obtain new results for many-player games (see discussion in
Subsection I-B). Furthermore, its proof is relatively simple,
and in particular "PCP-free". (See full version for details.)

The main challenge: locality.

Our ultimate goal is to construct a two-player game that
simulates the Brouwer function from Theorem I.4. This
is done via an imitation gadget: Alice’s mixed strategy
induces a point \( x^{(A)} \in [0, 1]^n \); Bob’s strategy induces
\( x^{(B)} \in [0, 1]^n \); Alice wants to minimize \( \| x^{(A)} - x^{(B)} \|_2 \),
whereas Bob wants to minimize \( \| f (x^{(A)}) - x^{(B)} \|_2 \). Alice
and Bob are both satisfied at a fixed point, where \( x^{(A)} =
\| x^{(B)} \|_2 \). Alice and Bob are both satisfied at a fixed point, where \( x^{(A)} =
\| x^{(B)} \|_2 \). Alice and Bob are both satisfied at a fixed point, where \( x^{(A)} =
\| x^{(B)} \|_2 \). Alice and Bob are both satisfied at a fixed point, where \( x^{(A)} =
\| x^{(B)} \|_2 \).

The main obstacle is that we want to incentivize Bob to
minimize \( \| f (x^{(A)}) - x^{(B)} \|_2 \) via local constraints (payoffs
- each depends on one pure strategy), while \( f (x^{(A)}) \) has a
global dependency on Alice’s entire mixed strategy.

Our goal is thus to construct a hard Brouwer function that
can be locally computed. How local does the computation
need to be? In a game of size \( 2^n \times 2^n \), each strategy can
faithfully store information about \( \sqrt{n} \) bits. Specifically, our
construction will be \( n^{1/2+o(1)} \)-local.

We haven’t yet defined exactly what it means for our
construction to be \( \approx n^{1/2+o(1)} \)-local; the exact formulation
is quite cumbersome as the query access needs to be partly
adaptive, robust to noise, etc. Eventually we formalize the
“locality” of our Brouwer function via a statement about
multiplayer games. On a high level, however, our goal is
to show that for any \( j \in \{1, \ldots, n\} \), the \( j \)-th output
\( f_j (x) \) can be approximately computed, with high probability, by
accessing \( x \) at only \( n^{1/2+o(1)} \) coordinates.

This is a good place to note that achieving any sense of
“local computation” in our setting is surprising, even if we
consider just the error correcting encoding for our Brouwer
function: in order to maintain constant relative distance, an
average bit of the output must depend on a constant fraction
of the input bits!

LocalEndoFALINE

In order to introduce locality, we go back to the ENDoFAL-
INE problem. “Wishful thinking”: imagine that we could
replace the arbitrary predecessor and successor circuits in
ENDofALINE with NC^0 (constant depth and constant fan-
in) circuits \( S^{\text{local}}, P^{\text{local}} : \{0, 1\}^n \to \{0, 1\}^n \), so that each
output bit only depends on a constant number of input bits.
Imagine further that we had the guarantee that for each
input, the outputs of \( S^{\text{local}}, P^{\text{local}} \) differ from the input
on just a constant number of bits. Additionally, it would be
really nice if we had a succinct pointer that immediately told
us which bits are about to be replaced. (We later call this
succinct pointer the counter, because it also cycles through
its possible values in a fixed order.)
Suppose all our wishes came true, and furthermore the hard Brouwer function from Theorem I.4 used a linear error correcting code. Then, we could use the encoding of the counter, henceforth $C(u)$, to read only the bits that are about to be replaced, and the inputs that determine the new values of those bits. Thus, using only local access to a tiny fraction of the bits ($|C(u)| + O(1)$), we can construct a difference vector $u - S_{\text{local}}(u)$ (which is 0 almost everywhere). As we discussed above, the encodings $E(u), E(S_{\text{local}}(u))$ must differ on a constant fraction of the bits - but because the code is linear, we can also locally construct the difference vector $E(u) - E(S_{\text{local}}(u)) = E(u - S_{\text{local}}(u))$. Given $E(u) = E(S_{\text{local}}(u))$, we can locally compute any bit of $E(S_{\text{local}}(u))$ by accessing only the corresponding bit of $E(u)$.

Back to reality: unfortunately we do not know of a reduction to such a restricted variant of ENDOFALINE. Surprisingly, we can almost do that. The problem LOCALENDOFALINE (formally defined in the full version) satisfies all the guarantees defined above, is linear-time reducible from ENDOFALINE, but has one caveat: it is only defined on a strict subset $V_{\text{local}}$ of the discrete hypercube ($V_{\text{local}} \subseteq \{0,1\}^n$). Verifying that a vertex belongs to $V$ is quite easy - it can be done in $AC^0$. Let us take a brief break to acknowledge this new insight about the canonical problem of PPAD:

**Theorem I.5.** The predecessor and successor circuits of ENDOFALINE are, wlog, $AC^0$ circuits.

The class $AC^0$ is quite restricted, but the outputs of its circuits are not local functions of the inputs. Now, we want to represent $u$ in a way that will make it possible to locally determine whether $u \in V_{\text{local}}$ or not. To this end we augment the linear error correcting encoding $E(u)$ with a probabilistically checkable proof (PCP) $\pi(u)$ of the statement ($u \in V_{\text{local}}$).

**Our holographic proof system**

Some authors distinguish between PCPs and holographic proofs\footnote{In a nutshell, PCPs or holographic proofs are proofs that can be verified “locally” (with high probability) by reading only a small (random) portion of the proof; see e.g. [7, Chapter 18] for many more details.}: a PCP verifier has unrestricted access to the instance, and queries the proof locally; whereas the holographic proof verifier has restricted, local access to both the proof (and an error correcting encoding of it). In this sense, what we actually want is a holographic proof.

We construct a holographic proof system with some very unusual properties. We are able to achieve them thanks to our modest locality desideratum: $n^{1/2+o(1)}$, as opposed to the typical polylog($n$) or $O(1)$. We highlight here a few of those properties; full version for details.

- **(Local proof construction)** The most surprising property of our holographic proof system is that the proof $\pi(u)$ can be constructed from local access to the encoding $E(u)$. In particular, note that we can locally compute $E(S_{\text{local}}(u))$ because $E(\cdot)$ is linear - but $\pi(\cdot)$ is not. Once we obtain $E(S_{\text{local}}(u))$, we can use local proof construction to compute $\pi(S_{\text{local}}(u))$ locally.

- **(Very low random-bit complexity)** Our verifier is only allowed to use $(1/2+o(1)) \log_2 n$ random bits - this is much lower even than the $\log_2 n$ bits necessary to choose one entry at random. In related works, similar random-bit complexity was achieved by bundling the entries together via “birthday repetition”. To some extent, something similar happens here, but our locality is already $n^{1/2+o(1)}$ so no bundling (or repetition) is necessary. To achieve nearly optimal random-bit complexity, we use $\lambda$-biased sets over large finite fields together with the Sampling Lemma of Ben-Sasson et al. [21].

- **(Tolerant verifier)** Typically, a verifier must reject (with high probability) whenever the input is far from valid, but it is allowed to reject even if the input is off by only one bit. Our verifier, however, is required to accept (with high probability) inputs that are close to valid proofs. (This is related to the notion of “tolerant testing”, which was defined in [58] and discussed in [42] for locally testable codes.)

- **(Local decoding)** We make explicit use of the property that our holographic proof system is also a locally decodable code. While the relations between PCPs and locally testable codes have been heavily explored (see e.g. Goldreich’s survey [41]), the connection to locally decodable codes is not as immediate. Nevertheless, related ideas of Locally Decode/Reject Codes [56] and decodable PCP [38] have been used before in order to facilitate composition of tests (our holographic proof system, in contrast, is essentially composition-free). Fortunately, as noted by [38] many constructions of PCPs are already implicitly locally decodable.

- **(Robust everything)** Ben-Sasson et al. [20] introduce a notion of robust soundness, where on an invalid proof, the string read by the verifier must be far from any acceptable string. (Originally the requirement is far in expectation, but we want far with high probability.) Another way of looking at the same requirement, is that even if a malicious prover adaptively changes a small fraction of the bits queried by the verifier, the test is still sound. In this sense, we require that all our guarantees, not just soundness, continue to hold (with high probability) even if a malicious entity adaptively changes a small fraction of the bits queried by the verifier.

How is local proof construction possible? At a high level, our holographic proof system expects an encoding of $u$ as a
low-degree t-variate polynomial, and a few more low-degree
 t-variate polynomials, that encode the proof of u ∈ VLOC.
 (This is essentially the standard “arithmetization”, dating
 back at least to [11], [63], although our construction is
 most directly inspired by [59], [65].) In our actual proof,
 t is a small super-constant, e.g. \( t \approx \sqrt{\log n} \), but for our
 exposition here, let us consider t = 2, i.e. we have bivariate
 polynomials.

The most interesting part of the proof verification is
 testing that a certain low-degree polynomial \( \Psi : G^2 \to G \),
 for some finite field \( G \) of size \(|G| = \Theta \left( \frac{1}{\log n} \right) \), is identically
 zero over all of \( F^2 \), for some subset \( F \subseteq G \) of cardinality
 \(|F| = |G|/\text{polylog}(n)\). This can be done by expecting the
 prover to provide the following low-degree polynomials:

\[
\Psi'(x, y) \triangleq \sum_{f_i \in F} \Psi(x, f_i) y^i
\]

\[
\Psi''(x, y) \triangleq \sum_{f_i \in F} \Psi'(f_j, y) x^i.
\]

Then, \( \Psi''(x, y) = \sum_{f_i, f_j \in F} \Psi'(f_j, f_i) x^i y^i \) is the zero
 polynomial if and only if \( \Psi \) is indeed identically zero over
 all of \( F^2 \). \( \Psi(x, y) \) can be computed by accessing \( E(u) \)
 on just a constant number of entries. Thus, computing \( \Psi(x, f_i) \)
 requires \( \sum_{f_i \in F} \Psi'(f_j, y) x^i = 0 \) for sufficiently many \( (x, y) \).

Putting it all together via polymatrix games

The above arguments suffice to construct a hard Brouwer
 function (in the sense of Theorem I.4) that can be computed
 “\( n^{1+o(1)} \)-locally”. We formalize this statement in terms of
 approximate Nash equilibria in a polymatrix game.

Definition I.6 (Polymatrix games). In a polymatrix game,
each pair of players simultaneously plays a separate two-
player subgame. Every player has to play the same strategy
in every two-player subgame, and her utility is the sum of
 her subgame utilities. The game is given in the form of the
 payoff matrix for each two-player subgame.

We construct a bipartite polymatrix game between
 \( n^{1+o(1)} \) players with \( 2^{n^{1+o(1)}} \) actions each. By “bi-
partite”, we mean that each player on Alice’s side only
 interacts with players on Bob’s side and vice versa. The
 important term here is “polymatrix”: it means that when
 we compute the payoffs in each subgame, they can only
 depend on the \( n^{1+o(1)} \) coordinates described by the two
 players’ strategies. It is in this sense that we guarantee “local
 computation”.

The mixed strategy profile \( \mathcal{A} \) of all the players on Alice’s
 side of the bipartite game induces a vector \( \mathbf{x}(\mathcal{A}) \in [0, 1]^m \),
 for some \( m = n^{1+o(1)} \). The mixed strategy profile \( \mathcal{B} \) of all
 the players on Bob’s side induces a vector \( \mathbf{x}(\mathcal{B}) \in [0, 1]^m \).

Our main technical result is:

**Proposition I.7** (Informal). If all but an \( \epsilon \)-fraction of the
 players play \( \epsilon \)-optimally, then \( \| \mathbf{x}(\mathcal{A}) - \mathbf{x}(\mathcal{B}) \|_2^2 = O(\epsilon) \) and
 \( \| f(\mathbf{x}(\mathcal{A})) - \mathbf{x}(\mathcal{B}) \|_2^2 = O(\epsilon) \).

Each player on Alice’s side corresponds to one of the
 PCP verifier’s random string. Her strategy corresponds to
 an assignment to the bits queried by the verifier given this
 random string. On Bob’s side, we consider a partition of
 \( \{ 1, \ldots, m \} \) into \( n^{1+o(1)} \) tuples of \( n^{1+o(1)} \) indices each.
 Each player on Bob’s side assigns values to one such tuple.

On each two-player subgame, the player on Alice’s side
 is incentivized to imitate the assignment of the player on
 Bob’s side on the few coordinates where they intersect.
 The player on Bob’s side, uses Alice’s strategy to locally
 compute \( f_j(\mathbf{x}(\mathcal{A})) \) on a few \( j \)'s in his \( n^{1+o(1)} \)-tuple of
 coordinates. This computation may be inaccurate, but we
 can guarantee that for most coordinates it is approximately
 correct most of the time.

**From polymatrix to bimatrix**

The final reduction from the polymatrix game to two-
player game follows more or less from known techniques
 for hardness of Nash equilibria [4], [32], [15]. We let each
 of Alice and Bob control one side of the bipartite polymatrix
 game. In particular, each strategy in the two-player game
 corresponds to picking a player of the polymatrix game, and
 a strategy for that player. We add a gadget due to Althofer
 [4] to guarantee that Alice and Bob mix approximately
 uniformly across all their players. See full version for details.

**B. Results for multiplayer relaxations of Nash equilibrium**

Our hardness for norm-2 approximate Brouwer fixed
 point (Theorem I.4) has some important consequences
 for multiplayer games. All our results in this regime (as well as
 much of the existing literature) are inspired by a paper of
 Babichenko [13] and a blog post of Shmaya [64].

For multiplayer games, there are several interesting ques-
tions one can ask. First, note that the normal form represen-
tation of the game is exponential in the number of players, so
 it is difficult to talk about computational complexity. To al-
leviate this, different restricted classes of multiplayer games
 have been studied. We have already seen polymatrix games
 (Definition I.6), which have a succinct description in terms of
 the normal forms of the \( \binom{n}{2} \) bimatrix subgames. Another
 interesting class is graphical games where we are given a
 (low-degree) graph over the players, and each player’s
 utility is only affected by the actions of its neighbors. The
 graph of the game constructed in Proposition I.7 has a high
 degree, but it is important that it is bipartite, as are all
 the games described below. Most generally, we can talk
 about the class of succinct games, which are described via
a circuit that computes any entry of the payoff tensors (this includes polymatrix and graphical games). Finally, there has been recent significant progress on the query complexity of finding approximate Nash equilibria in arbitrary **n**-player games where the payoff tensors are given via a black-box oracle [14], [43], [13], [28].

There are also a few different notions of approximation of Nash equilibrium. The strictest notion, **ε**-Well-Supported Nash Equilibrium, requires that for every action in the support of every player, the expected utility, given other players’ mixed strategies, is within (additive) **ε** of the optimal strategy for that player. For this notion, Babichenko [13] showed a $2^{Ω(n)}$ lower bound on query complexity for any (possibly randomized) algorithm. In followup work, [60] showed PPAD-completeness for succinct games, and soon after [62] extended this PPAD-completeness to games that are both polymatrix degree-3-graphical.

The most central model in the literature, **ε**-Approximate Nash Equilibrium, requires that every player’s expected utility from her mixed strategy is within **ε** of the optimum she can achieve (given other players’ strategies). I.e., any player is allowed to assign a small probability to poor strategies, as long as in expectation she does well. The last PPAD-completeness result extends immediately to this model (the two notions of approximation are equivalent, up to constant factors, for constant degree graphical games). For query complexity, Hart and Nisan [43] and Babichenko [13] asked whether the latter’s exponential lower bound can be extended to **ε**-Approximate Nash Equilibrium. Very recently Chen et al. [28] solved it almost entirely, showing a lower bound of $2^{Ω(n/\log n)}$; and they asked whether the $Θ(\log n)$ gap in the exponent can be resolved. Here, we obtain a tight $2^{Ω(n)}$ lower bound on the query complexity, as well as stronger inapproximability guarantees.

Finally, the most lenient notion is that of **(ε, δ)**-WeakNash: it only requires that a $(1−δ)$-fraction of the players play **ε**-optimally (“Can almost everybody be almost happy?”). This notion was recently defined in [15] who conjectured that it is also PPAD-complete for polymatrix, graphical games. Their conjecture remains an interesting open problem (see Subsection I-C). Here, as a consequence of Theorem I.4, we prove that **(ε, δ)**-WeakNash is PPAD-complete for the more general class of succinct games.

**Corollary I.8.** There exist constants **ε**, **δ** > 0, such that finding an **(ε, δ)**-WeakNash is PPAD-hard for succinct multiplayer games where each player has two actions.

Furthermore, as we hinted earlier, our proof also extends to giving truly exponential lower bounds on the query complexity:

**Corollary I.9.** There exist constants **ε**, **δ** > 0, such that any (potentially randomized) algorithm for finding an **(ε, δ)**-WeakNash for multiplayer games where each player has two actions requires $2^{Ω(n)}$ queries to the players’ payoffs.

**C. The PCP Conjecture for PPAD**

Rather than posing new open problems, let us restate the following conjecture due to [15]:

**Conjecture I.10** (PCP for PPAD; [15]). There exist constants **ε**, **δ** > 0 such that finding an **(ε, δ)**-WeakNash in a bipartite, degree three polymatrix game with two actions per player is PPAD-complete.

The main original motivation was an approach to prove our main theorem given this conjecture. As pointed out by [15], it turns out that resolving this conjecture would also have interesting consequences for relative approximations of two-player Nash equilibrium, as well as applications to inapproximability of market equilibrium.

More importantly, this question is interesting in its own right: how far can we extend the ideas from the PCP Theorem (for NP) to the world of PPAD? The PCP [r(n), q(n)] characterization [8] is mainly concerned with two parameters: r(n), the number of random bits, and q(n), the number of bits read from the proof. A major tool in all proofs of the PCP Theorem is verifier composition: in the work of Polishchuk and Spielman [59], [65], for example, it is first shown that NP ⊆ PCP [O(log n), n^{1/2+δo(1)}], and then via composition it is eventually obtained that NP = PCP [O(log n), O(1)]. In some informal sense, one may think of our main technical result as something analogous to PPAD ⊆ PCP [O(1/2 + o(1)) log n, n^{1/2+δo(1)}]. Furthermore, our techniques build on many existing ideas from the PCP literature [12], [59], [65], [21], [20] that have been used to show similar statements for NP. It is thus natural to ask: is there a sense in which our “verifier” can be composed? can such composition eventually resolve the PCP Conjecture for PPAD?

More generally, some of the tools we use here, even as simple as error correcting codes, have been the basic building blocks in hardness of approximation for decades, yet to the best of our knowledge have not been used before for any problem in PPAD. We hope to see other applications of similar ideas in this regime.

**D. Additional related work**

Aside from [15], previous attempts to show lower bounds for approximate Nash in two player games have mostly focused on limited models of computation [35] and lower bounding the support required for obtaining approximate equilibria [4], [39], [6], [5] (in contrast, [54]’s algorithm runs

---

5We stress that our analogy is very loose. For example, we are not aware of any formal extension of PCP to function problems, and it is well known that NP ⊆ PCP [(1/2 + o(1)) log n, n^{1/2+δo(1)}].

6In fact, in the few months since our paper first appeared, our techniques already found applications for lower bounds on the communication complexity of Nash equilibrium [16].
in quasi-polynomial time because there exist approximate equilibria with support size at most $O\left(\frac{\log n}{\epsilon^2}\right)$.

**Birthday repetition and related quasi-polynomial lower bounds:** Hazan and Krauthgamer [44] showed that finding an $\epsilon$-Approximate Nash Equilibrium with $\epsilon$-optimal welfare is as hard as the PLANTED-CLIQUE problem; Austrin et al. [9] later showed that the optimal-welfare constraint can be replaced by other decision problems. Braverman et al. [25] recently showed that the hardness PLANTED-CLIQUE can be replaced by the Exponential Time Hypothesis, the NP-analog of the ETH for PPAD we use here. The work of Braverman et al, together with an earlier paper by Aaronson et al. [1] inspired a line of works on quasi-polynomial hardness results via the technique of “birthday repetition” [15], [24], [61], [22]. In particular [15] investigated whether birthday repetition can give quasi-polynomial hardness for finding any $\epsilon$-Approximate Nash Equilibrium (our main theorem). As we discussed in Subsection I-C the main obstacle is that we don’t have a PPAD-analogue for the PCP Theorem.

**Multiplicative hardness of approximation:** Daskalakis [31] and our recent work [60] show that finding an $\epsilon$-relative Well-Supported Nash Equilibrium in two-player games is PPAD-hard. The case of $\epsilon$-relative Approximate Nash Equilibrium is still open: our main theorem implies that it requires at least quasi-polynomial time, but it is not known whether it is PPAD-hard, or even if it requires a large support (see also discussion in [15]).

**Approximation algorithms:** The state of the art for games with arbitrary payoffs is $\approx 0.339$ for two-player games due to Tsaknakis and Spirakis [67] and $0.5 + \epsilon$ for polymatrix games due to Deligkas et al. [37]. For two-player games, PTAS have been given for the special cases of constant rank games by Kannan and Theobald [48], small-probability games by Daskalakis and Papadimitriou [35], positive semi-definite games by Alon et al. [3], and sparse games by Barman [17]. For games with many players and a constant number of strategies, PTAS were given for the special cases of anonymous games by Daskalakis and Papadimitriou [36] and polymatrix games on a tree by Barman et al. [18]. Finally, let us return to the more general class of succinct $n$-player games, and mention an approximation algorithm due to Goldberg and Roth [40]; their algorithm runs in exponential time, but uses only a polynomial number of oracle queries.

**Communication complexity:** The hard instance of Brouwer we construct here (Theorem I.4) has already been useful in followup work [16], for proving lower bounds on the communication complexity of approximate Nash equilibrium in $N \times N$ two-player games, as well as binary-action $n$-player games.

**ACKNOWLEDGMENT**

I am grateful to Yakov Babichenko, Jonah Brown-Cohen, Karthik C.S., Alessandro Chiesa, Elad Haramaty, Christos Papadimitriou, Muli Safra, Luca Trevisan, Michael Viderman, and anonymous reviewers for inspiring discussions and suggestions. This research was supported by Microsoft Research PhD Fellowship. It was also supported in part by NSF grant CCF1408635 and by Templeton Foundation grant 3966. This work was done in part at the Simons Institute for the Theory of Computing.

**REFERENCES**


