Settling the Complexity of Nash Equilibrium in Congestion Games

Yakov Babichenko*
yakovbab@technion.ac.il
Technion, Israel institute of Technology
Haifa, Israel

Aviad Rubinstein†
aviad@cs.stanford.edu
Stanford University
Stanford, California, USA

ABSTRACT

We consider (i) the problem of finding a (possibly mixed) Nash equilibrium in congestion games, and (ii) the problem of finding an (exponential precision) fixed point of the gradient descent dynamics of a smooth function $f : [0, 1]^n \to \mathbb{R}$. We prove that these problems are equivalent. Our result holds for various explicit descriptions of f, ranging from (almost general) arithmetic circuits, to degree-5 polynomials. By a very recent result of [Fearnley et al., STOC 2021], this implies that these problems are PPAD ∩ PLS-complete. As a corollary, we also obtain the following equivalence of complexity classes:

$$\mathsf{CCLS} = \mathsf{PPAD} \cap \mathsf{PLS}.$$

CCS CONCEPTS

• **Theory of computation** → **Complexity classes**; **Nonconvex optimization**; **Exact and approximate computation of equilibria**.

KEYWORDS

Equilibrium computation, computational complexity, gradient descent, potential games, congestion games

ACM Reference Format:
Yakov Babichenko and Aviad Rubinstein. 2021. Settling the Complexity of Nash Equilibrium in Congestion Games. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3406325.3451039

1 INTRODUCTION

1.1 Game Theory

Nash equilibrium is the central solution concept in Game Theory. In recent decades, its universality as a solution concept has come under a computational critique: Finding a Nash equilibrium is, in general, a computationally intractable problem [20, 26]; and if centralized, specially designed algorithms cannot find an equilibrium, how can we expect decentralized, selfish agents to converge to one?

Identifying classes of games where Nash equilibrium is tractable is an important open problem (e.g. [25]). Congestion games [55] are arguably the most important class of games for computer scientists, capturing useful applications like resource sharing and routing in networks. They play a fundamental role in Algorithmic Game Theory since the inception of the field two decades ago. In particular, a celebrated[1] research program on Price of Anarchy has focused on characterizing the outcomes of these games at equilibrium. But an important question remains about this modeling assumption:

> Do agents in congestion games play equilibrium strategies?

Again, if we're hoping for selfish, decentralized agents to converge to an equilibrium, at the very least an algorithm should be able to find one.

At the source of computational hardness for general games is the combinatorial number of *mixed* strategies that the algorithm has to consider. In contrast, congestion games admit pure equilibria, which makes them a prime candidate for tractable equilibrium computation. The seminal work of [34] (see also [2, 62]) ruled out the most immediate application of this idea: while pure equilibria exist for congestion games, finding them is again intractable. But more generally, it is natural to hope that the special structure of congestion games allows for efficient computation of (possibly mixed) Nash equilibria.

1.2 Optimization

Our paper also touches on a fundamental problem in continuous optimization: Minimize a continuous function

$$f : [0, 1]^n \to [0, 1].$$

Clearly we cannot hope to find a global minimum, but, under a mild smoothness assumption (see Remark 1), there is a useful and total local solution concept: (approximate) fixed points of the gradient descent dynamics ("KKT points"). By a simple potential argument, gradient descent itself always converges to such ϵ-approximate fixed point in $\mathrm{poly}(1/\epsilon)$ iterations. But in some cases, e.g. when f is also convex, algorithms like the center of gravity method converge in $\mathrm{poly}(n, \log(1/\epsilon))$ iterations [12, Theorem 2.1]. For what nonconvex functions can we achieve exponentially fast (in $1/\epsilon$) convergence? (Indeed, accelerated methods make it possible for some nonconvex functions, e.g. [16, Theorem 6.3].)

*Supported by the Binational Science Foundation award (BSF #2018397) and the Greman-Israel Foundation for Scientific Research and Development (GIF #2027111).
†Supported by a David and Lucile Packard Fellowship.

[1] Including well-deserved citations for prizes named after Godel, Tucker, von Neumann (twice), and Kalai.

In the worst case, when f is given as a black-box oracle, unconditional query complexity results (e.g. [13, 17, 64]) rule out exponential convergence. *When f has a succinct representation, e.g. as a sum of monomials, query complexity arguments can only rule out exponential convergence for specific algorithms.* To address the possibility of exponential speedup by general algorithms, we need to turn to computational complexity. In this paper, we study the computational problem GD-FixedPoint (see Meta Definition 1), where the goal is to find an approximate fixed point of the gradient descent dynamics. We consider 4 variants of this problem, where the potential f is succinctly represented assuming any of the following different structures:

(1) In the most general case, f is succinctly described by an arithmetic circuit[2]
(2) When defining the class CCLS, [27] consider the case where f is guaranteed[3] to be component-wise convex.
(3) [27] consider the restriction that f is explicitly given as a sum of monomials (note that this is orthogonal to the component-wise convexity condition). This problem is called KKT in [27].
(4) In the most restrictive case, f is a sum of component-wise convex monomials of constant degree.

Remark 1. For all of these cases we make the assumption that f has a finite (but possibly exponentially large) Lipschitz constant of the gradients. We call this Lipschitz constant the *smoothness parameter.*

1.3 Computational Complexity

From a technical perspective, the problem Congestion of finding a Nash equilibrium in congestion games, and the aforementioned GD-FixedPoint, belong to a special class, TFNP, of *total* search problems, i.e. ones that always have a solution. The aforementioned works prove that finding a Nash equilibrium in a general game is complete for the class PPAD, which captures fixed point computations for *continuous functions.* Finding a pure equilibrium in congestion games is complete for the class PLS, which captures local maximum computation for *discrete potential funcions.* Finding a Nash equilibrium in congestion games is a special case of both problems, i.e. it lies in PPAD ∩ PLS, and hence unlikely to be complete for either class[4].

Since the seminal paper of [27], the class PPAD ∩ PLS has been at the frontier of research on total search problems (e.g. [30, 32, 35–38, 42–44]). This research direction is particularly exciting because these problems lie quite low in the TFNP hierarchy, so a-priori any of them could plausibly admit a polynomial time algorithm. Several natural problems have been identified, and a few complexity sub-classes have been defined (including CLS[5], CCLS, EOPL, and UniqueEOPL). Yet, until this paper, neither PPAD ∩ PLS nor any of its sub-classes were known to have any natural[6] complete problems.

1.4 Our Results

Theorem (Main result, informal).
Congestion *and all 4 variants of* GD-FixedPoint *are computationally equivalent.*

In the context of optimization, it immediately follows from our main theorem that local maximization (in the GD-FixedPoint sense) of an arbitrary smooth function can be reduced to an optimization of a degree 5 polynomial. Very recently[7] [35] proved that GD-FixedPoint is PLS ∩ PPAD-complete. We therefore obtain the following corollaries:

Corollary (Complexity consequences).

- Congestion *and all 4 variants of* GD-FixedPoint *are* PLS ∩ PPAD*-complete.*[8] *This is the first case of a natural*[9] *complete problem for* PLS ∩ PPAD *or any of its subclasses.*
- CCLS = PLS ∩ PPAD.

1.5 Related Work

Our proof combines ideas from the two long lines of work on PLS-completness (e.g. [1, 2, 34, 46, 48, 50, 61, 62]) and PPAD-completeness (e.g. [18–21, 23, 26, 31, 41, 51, 53, 60]). It also uses ideas from our recent paper on the *communication* complexity of Nash equilibrium in potential and congestion games [8].

Recently, [29] also study a variant of the problem GD-FixedPoint. The focus of their paper is on $1/\text{poly}(n)$ or even constant approximation, whereas we seek exponentially precise approximation. Note that for GD-FixedPoint, it is easy to compute a $1/\text{poly}(n)$-approximate fixed point in polynomial time because the potential is bounded. In contrast, for the gradient descent-ascent dynamics (GDA-FixedPoint) the potential may increase and decrease alternatingly, which makes the problem much harder for approximation. Indeed, for an even harder variant of approximate GDA-FixedPoint where the domain is not a cartesian product, [29] prove that the problem is PPAD-complete. (Approximate GDA-FixedPoint over e.g. the hypercube remains an exciting open problem.)

Historically, the first Price of Anarchy (PoA) results for congestion games focused on pure equilibrium (e.g. [24, 49, 57]), and were then extended to mixed Nash equilibrum (e.g. [3, 5, 65]). The barriers to equilibrium computation in congestion games inspired the development of techniques, in particular Roughgarden's smoothness framework, for bounding the PoA with respect to tractable

[2]Formally we need to require some restrictions on this arithmetic circuit to avoid doubly-exponentially large numbers by repeated squaring, see [28, 35] for details.

[3]Formally to avoid promise problems the algorithm may return a certificate that the component-wise convexity is violated. In contrast, for the fourth variant it is easy to tell if a monomial is component-wise convex.

[4]PPAD and PLS are generally believed to be incomparable, e.g. due to oracle separations [14, 15].

[5]Very recent breakthrough shows that in fact CLS = PPAD ∩ PLS [35]!

[6] In the sense that all known CLS-complete [30, 36], UniqueEOPL-complete [37], and PPAD ∩ PLS-complete [35] problems are described in terms of an oracle function succinctly represented by a circuit. From an algorithm-designer point of view, it is typically more natural -and often much easier- to study the *query complexity* of such oracle problems. (The computational complexity of oracle problems is still very interesting, e.g. as a stepping stone for understanding natural problems!) See further discussions in [39, 63].

[7]In fact, [35] was developed concurrently and independently to a previous version of our paper. The proofs in this version (while still unfortunately long) have been simplified significantly since [35] show that GD-FixedPoint is already hard in only two dimensions.

[8]The completeness for the first variant of GD-FixedPoint was proved by [35].

[9]See Footnote 6.

solution concepts such as correlated equilibrium (e.g. [10, 56]). Our results show that mixed Nash equilibrium in congestion game is intractable, further motivating the latter line of work on analyzing tractable solution concepts.

Another interesting approach to circumventing the computational barriers to computing pure equilibrium in congestion games considers beyond worst case analysis. Specifically a recent line of works applies smoothed analysis to the special case of polymatrix identical interest games [4, 9, 11, 22, 33]. It is an interesting open question whether considering mixed Nash equilibrium can improve those results. Another interesting question is whether this analysis can extend to more general congestion games (for an appropriate definition of smoothing).

2 TECHNICAL HIGHLIGHTS

In this section we exposite some techincal highlights from the proof of our main reduction. Specifically, we reduce the PLS ∩ PPAD-complete 2D-GD-FixedPoint to finding a Nash equilibrium in a sub-class of congestion games that we now introduce.

Polytensor identical-interest games. We consider a class of games that satisfies two orthogonal constraints:

- 5-**Polytensor**[10] **games:** The utility of Player i is given as a sum of sub-utilities, each depending on the strategies of only 5 players (including i).
- **Identical interest**[11] **games:** All players have the same utility function.

Remark 2. In identical interest 5-polytensor games the local sub-utilities have identical interest. Note that this does not create an identical-interest game (because every player has different local interactions). However, we can alternatively consider a strategically equivalent game where every player gets the sum of *all* sub-utilities in *all* local interactions (including those where she does not participate). This latter game is an identical interest game. The sets of Nash equilibria in these two games coincide.

We reduce 2D-GD-FixedPoint to finding a Nash equilibrium in the class of *polytensor identical-interest games*, i.e. games that satisfy both desiderata. Our primary motivation for studying this class of games is as an intermediary problem for our reductions, but it may also be of independent game-theoretic interest (e.g. it is a natural generalization of the popular class of polymatrix identical-interest games [4, 9, 11, 22, 27, 33]). See Section 3.1.2 for details.

2.1 Potential+Imitation Game

Our reduction begins with a potential function $\phi : [0,1]^2 \to [0,1]$, which we will try to *maximize*.[12] Consider the 2-player infinite-actions potential game where Player $j = 1, 2$ chooses a value for coordinate $x_j \in [0,1]$, and every player's utility is $\phi(x)$. In any pure equilibrium, the players collectively choose a point $x \in [0,1]^2$ where any unilateral change to one coordinate, in particular an infinitesimal change, does not improve the potential — hence in particular this is a fixed point of the gradient descent dynamics. Before we talk about reducing the number of actions, there is an infamous technical obstacle around dealing with *mixed* strategies: a mixed strategies of the players need not correspond to any particular point x, let alone one that is a fixed point of the gradient descent dynamics.

Challenge 1. Mixed strategies need not correspond to any particular point.

It is not possible to completely rule out mixed strategies, but we will incentivize players to choose strategies that correspond to x_j's that are very tightly concentrated. If the potential function satisfies a mild smoothness condition (see Remark 1), we can guarantee that the gradient is approximately constant in any sufficiently small neighborhood. Therefore if the mixed x's are supported in a neighborhood that does not contain an approximate gradient descent fixed point, there is some direction where the gradient always shows an improving deviation.

To force the mixed strategies to concentrate, we use the *imitation game*, which has been the driving force in recent advances on complexity of Nash equilibrium (e.g. [6, 7, 40, 45, 58–60], and in particular the authors' own recent work on communication complexity of Nash equilibrium in potential games [8]. Specifically, we add 2 additional players who choose a second point $y \in [0,1]^2$, and update the identical utility function to

$$U(x, y) := -\|x - y\|_2^2 + \epsilon\phi(x), \tag{1}$$

for a sufficiently small $\epsilon > 0$.

First, since the main incentive of each player is to minimize the square distance from the opponent the best-reply against any *mixed* strategy of the opponent is ϵ-close to the expectation of the opponent's strategy. Mutual imitation of both players implies that their (mixed) strategies are supported ϵ-close to each other.[13] If in this region the gradient in some direction is positive (and does not get out of the region $[0,1]^2$), then the x-player has an incentive to mildly deviate toward this direction: her imitation loss is quadratic while her improvement in the gradient is linear. For sufficiently small deviation the linear term is more significant and hence it cannot be an equilibrium.

This trick allows us to focus on pure strategies (for now - mixed strategies will come back to haunt us throughout the proof). We now move to reduce the number of actions.

2.2 Strategic Binary Representation and Veto Players

Challenge 2. We are interested in games with a polynomial number of actions for each player.

[10]These games are sometimes called *hypergraphical games* following the seminal [54], but we believe that the name polytensor better reflects the fact that they generalize *polymatrix games* [66] (which would be 2-polytensor games).

[11]In our context these are equivalent to *potential* games, and sometimes they're also called *coordination* games.

[12]There is somewhat of a discrepancy between the optimization literature which mostly talks about minimization and game theory where players try to maximize the potential function. We side with the game theorists here, but of course the gradient descent dynamics should be applied to $-\phi$.

[13]This concentration of mixed strategies, although intuitive, is the most challenging part of our proofs. The formal proof contains quite a few intermediate steps. See full version for details.

To obtain a finite game, we discretize the real numbers to finite precision. But we need to work with exponential precision (approximations up to $\pm 1/\text{poly}(n)$ are trivial), and so can no longer afford to let a single player represent each coordinate.

We consider binary representation of each $x_j, y_j \in [0,1]$ and we replace a single player who chooses a real number by a polynomial number of *bit-players*, each chooses a single bit in the binary representation. This modification is insufficient: if x_j wants to move infinitesimally from the number 011...1 to the number 100...0 (as is indeed the case in the potential+imitation game (1)) this requires a *joint* switching of actions of all bit players. In particular, a bad equilibrium may arise where players are playing 011...1, the gradient in this coordinate is positive, but no bit-player wants to switch her action unilaterally.

Challenge 3. We are interested in a binary representation where there is always a player who can unilaterally shift the number infinitesimally in either direction.

Over the last few years we've tried several different approaches to overcoming this obstacle. For example, one natural idea is to use the Gray code (or variants) where every two consecutive strings differ by only one bit. The issue with this approach is that it greatly complicates bit-player utilities: for the least-significant-bit-player to know in which direction it prefers to go it must know the parity of all other bit players for that coordinate. (And things get even messier when those more significant bit players use mixed strategies...)

We overcome this obstacle by introducing an additional *veto player*: the veto player has the power to impose a veto on any suffix of the binary string. The possible vetoes are 011...1 and 100...0. If the veto player imposes such a veto then the actions of the bit players in the suffix are essentially ignored and, instead, the veto action of the veto player is counted. This resolves the problematic issue above: if the x-players are playing 011...1 then they can move to 100...0 by a unilateral deviation of the veto player. The veto player has an additional important role in our reduction: it has an incentive to veto bit players who are playing mixed (because those mixed strategies cause a high imitation loss). So the existence of the veto player allows us to argue that the bits of x will essentially be pure.[14]

The next problem that arises with the basic imitation+potential game (1) is that in a "natural" problem the utility should not be specified by the arithmetic circuit computing $\phi()$.

2.3 Circuit Players: Computing Potential and Gradients

Challenge 4. The utilities of players in our game should be explicitly given, i.e. we cannot use the term $\phi(x)$ in the utility.

Similarly to the existing literature on the complexity of Nash equilibrium (see e.g., [20, 26, 60]) in general games (i.e., not potential games), instead of writing the potential ϕ in the utility we introduce *circuit players* that implement the circuit gates and are responsible for computing ϕ.

[14]For this task of vetoing players that are playing mixed we need to mildly enrich the set of possible vetoes. In addition to 011...1 and 100...0 the veto player can choose also the vetoes 00...0, 11...1, 0011...1 and 1100...0. It turns out that for our reduction this enrichment is sufficient.

Remark 3. Unlike [20, 26, 60] our circuit players have identical interest both with their predecessors and their successors in the circuit. We set the utility in the interaction with predecessors higher than the interaction with predecessors. This incentivizes each circuit player to match its bit to the correct computation of the predecessors (rather than adjusting her input in a way that her successor's action will be correct). This results in an exponential decay of the utilities in the depth of the circuit. In particular, the computed potential ϕ in the output has exponentially lower weight in the common utility relative to the input.

Circuit players allow us to correctly compute the potential, but that comes with the cost of separating the potential $\phi(x)$ from the players who determine x:

Challenge 5. Before we introduced circuit players, the gain of deviation from x to x' causes an immediate reward once the potential increases from $\phi(x)$ to $\phi(x') > \phi(x)$. Namely, when x-players switch from x to x' they gain the increase in the potential. Once we introduce circuit players, the x-players will not get the potential gain until the circuit players will update their actions and will calculate $\phi(x')$ instead of the currently computed $\phi(x)$. So how should we incentivize the x-players to unilaterally deviate in the direction of the gradient?[15]

In our reduction, those that will be responsible for deviation toward better potential are the y-players. To give them immediate reward for moving toward better potential we modify the circuit to calculate $\nabla_j \phi(x)$ too. We add to the utility of the y-team the terms $\epsilon y_j \nabla_j \phi(x)$ (for $j = 1, 2$). Whenever $\nabla_j \phi(x) > 0$ the y-team has an incentive to increase y_j. Whenever $\nabla_j \phi(x) < 0$ the y team has an incentive to decrease y_j. Once the y-team moves toward the gradient, the x-team has an incentive to move there too because it tries to imitate y.

The next challenge in the reduction is to bootstrap this tiny incentive for y-players to deviate slightly in the gradient direction to a large movement of x-players.

2.4 Guide Player and Sampling Gadgets

Challenge 6. When the x-team moves, it should flip the input to the circuit, causing the first gate to pay for wrong computation. This force is exponentially more significant than the force of trying to imitate y that have mildly moved toward the better potential in the direction of the gradient.[16]

We resolve this obstacle by introducing three modifications. First, we let the x-team and the y-team choose real numbers with precision that is much finer than the bits that influence the circuit's input. We denote by N_{in} the precision of the circuit's input, where the total number of bits for the x-team is $K \gg N_{in}$. This allows the x-team to imitate the move of the y-team in the direction of the gradient whenever x *is far from a multiple of* $2^{-N_{in}}$. If x is located

[15]Moreover, such a deviation creates a loss in the circuit computations. This issue will be discussed in Challenge 6.

[16]Correct circuit computations has exponentially higher weight than the computed gradient (see e.g. Remark 3). The y-team moves toward the gradient in a step size that is proportional to the gradient. Therefore, indeed the force of maintaining correct input computations is more significant.

close to the $2^{-N_{in}}$-grid, the x-team does not want to change the first N_{in} bits due to the reason of Challenge 6.

The second modification applies the *sampling technique* from the PPAD-literature. This technique was first introduced for 3-D in [26] (who called it the "averaging maneuver"), and later adapted to high dimensions by [20]. The basic idea is that instead of having a single x-team we introduce many copies for the x-team. Each team tries to imitate a different *shift* of the y-team. We design the shits to ensure that if one x-team is located close to the $2^{-N_{in}}$-grid then the other teams necessarily are located far from the grid.

Challenge 6 deals with an exponential gap between the force pushing towards the gradient and the force resisting flipping input bits for the circuit, so taking a uniform average over a polynomial number of samples cannot resolve this issue. Instead, our third modification introduces a gadget that carefully and adaptively assigns weights to the samples. This gadget is inspired by a classic PLS-completeness proof of [50]. Specifically, we add a *guide player*, whose role is to choose a "well behaved" sample of x: a sample that performs accurate imitation, computes the potential correctly, and has a relatively high value of the potential. The circuit computation of a particular x-sample is given high weight only if it is chosen by the guide player. Otherwise, if an x-sample is not chosen by the guide player, its circuit computation weight is negligible (relative to the incentive to imitate y). Our desire is that these problematic x-samples that are close to the boundary in one of the coordinates will not be chosen by the guide player, which will allow the x-sample to change the first N_{in} bits to accurately imitate the y-team (with a shift). Thereafter, the circuit players will adjust their computation to the new input, and then the guide player could choose this x-sample again.

The intuition for guide player's avoidance from choosing close-to-grid problematic samples is as follows. We compare the utility for the guide player in the problematic sample $x_{i,j}$ with her utility in a well behaved sample $x_{i',j}$ that is located far from the grid. The imitation in $x_{i',j}$ sample is very precise because the phenomenon of avoiding switching bits due to wrong circuit computations does not occur when the imitation target is far from the grid: an infinitesimal movement toward the imitation target does not change the first N_{in} bits. Circuit's computation in $x_{i',j}$ is also correct because the first N_{in} input bits are deterministic. The only term which, in principle, might be inferior for $x_{i',j}$ is the potential term. However, this cannot be the case either: assume that $\nabla_j \phi(x) > 0$. In such a case y_j has an incentive to (mildly) *over-imitate* her target $\hat{x}_j$. If the problematic team $x_{i,j}$ is located from the left side of the grid-point this means that the potential that is computed there is *weakly lower* than the potential in all other samples, and in particular, the potential at $x_{i',j}$ is either identical or better than in the problematic sample; see Figure 1 (a). If, on the other hand, the problematic team of $x_{i,j}$ is located from the right side of the grid-point this means that the imitation target $\hat{y}_{i,j}$ is also located from the right side the grid-point (recall that y_j over-imitates $\hat{x}_j$). In such a case x_j can smoothly move toward her imitation target $\hat{y}_{i,j}$ without changing the first N_{in} bits. See Figure 1(b).

3 DEFINITIONS AND PRELIMINARIES

In this section we formally define the computational problems that will be discussed in this paper. Along the definitions we provide some (simple) observations.

3.1 Game Theoretic Definitions

Our main interest game-theoretic object in this paper is the class of (explicit) congestion games with a polynomial number of facilities and actions for each player. We show that, in terms of complexity of finding a Nash equilibrium, this class is equivalent to c-polytensor identical interest games (a generalization of polymatrix identical interest games[17]). We now formally introduce each of this classes.

3.1.1 Congestion Games. In congestion games (see [52, 55]) we have a set of facilities F. Each player i has a collection of subsets $S^i_1, ..., S^i_{m_i} \subset F$ which are her actions. Given an action profile $S = (S^1, ..., S^n)$ the congestion on each facility f is defined to be the number of players that use this facility (i.e., $c_f(S) = |\{i : f \in S^i\}|$). Each facility has a cost function $l : [n] \to \mathbb{R}$ and the utility of player i is defined to be $u_i(S^1, ..., S^n) = \sum_{f \in S^i} l(c_f(S))$; namely, each player pays the total costs of all facilities she has chosen. In *explicit* congestion games the possible actions of players (i.e., $S^i_1, ..., S^i_{m_i} \subset F$) are given explicitly.

The problem Congestion gets as an input an explicit congestion game and an ϵ_N and outputs an ϵ_N-Nash equilibrium of the game.

3.1.2 Polytensor Identical Interest Games. An n-player *polymatrix game* (see [47, 66]) is given by a tuple of two-player utility functions $(u^i_{i,j}(a_i, a_j))_{i,j \in [n]}$ and the utility of player $i \in [n]$ is the sum of these two-player utilities $u^i(a_i, a_{-i}) = \sum_{j \in [n]} u^i_{i,j}(a_i, a_j)$. In the case where $u^i(a_i, a_j) = u^j(a_i, a_j)$ for every i, j we denote this term by $u^{i,j}(a_i, a_j)$.

In a generalization of polymatrix games that we call *c-polytensor games*, instead of two-player interactions (i.e., the local interaction can be described by a matrix), we allow $\leq c$-player interactions (i.e., local interactions can be described by a tensor of order $\leq c$). Namely, a c-polytensor game is given by a tuple $(u^i_S(a_S))_{i \in [n], S \subset [n], |S| = c, i \in S}$ where a_S denotes the action profile of the players in S. Similarly to polymatrix games the utility of a player is given by the sum of all her interactions: $u^i(a) = \sum_{S \subset [n], |S| = c, i \in S} u^i_S(a_S)$. In the case where $u^i(a_S) = u^j(a_S)$ for every $i, j \in S$ we denote this term by $u^S(a_S)$. In case where the number of actions for each player is bounded by m note that the representation size of a c-polytensor game is $O(n^{c+1} m^c)$. In particular, if c is a constant and $m = poly(n)$ then c-polytensor games admit a succinct representation.

The problem c–Polytensor-IdenticalInterest gets as an input a c-polytensor identical interest game and an ϵ_N and outputs an ϵ_N-Nash equilibrium of the game.

Our primary purpose for studying the problem Polytensor-IdenticalInterest is as an intermediary problem for our reductions, but its complexity may also be of independent game theoretic interest.

We argue that every c-polytensor identical interest game for a constant c is also an explicit congestion game. The set of facilities

[17] Also called *network coordination games* in [27].

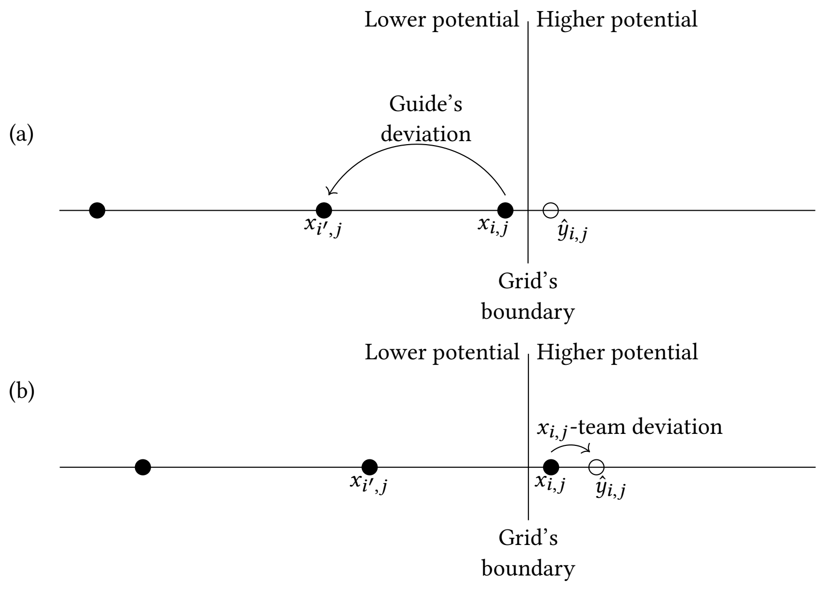

Figure 1: Figure 1(a) demonstrates the profitable deviation of the guide player in case the problematic sample $x_{i,j}$ is located from left side of the boundary. Figure 1(b) demonstrates the profitable deviation of $x_{i,j}$ team in case this problematic sample is located from right side of the boundary.

is $\{a_S : S \subset [n], |S| = c, a_i \in A_i$ for every $i \in S\}$ where A_i denotes player's i action set. By choosing an action $a_i \in A_i$ player i chooses the set of facilities $\{(a_i, a_{S'}) : S' \subset [n], |S'| = c - 1\}$; namely she chooses all facilities where her action is relevant (i.e., $i \in S$) and her action is a_i. In congestion games we should assign a common utility for all players that is a function of *the number* of users of each facility. In this case we set the common utility of each facility a_S to be $u^S(a_S)$ if it has c users; and 0 otherwise. Note that indeed only the "true" actions a_S have non-zero utility and indeed after summing the utilities over all facilities we obtain exactly the common utility of the c-polytensor identical interest game $\sum_{S\subset[n],|S|=c} u^S(a_S)$. Hence, the class of c-polytensor identical interest games is essentially a subclass of explicit congestion games.[18] In particular, we have the following:

Corollary 1. *For every constant c we have*

$$c - \textsc{Polytensor-IdenticalInterest} \leq \textsc{Congestion}.$$

3.1.3 Nash Equilibrium. A mixed action profile $(x_i)_{i\in[n]}$ where $x_i \in \Delta(A_i)$ is an ϵ-*Nash equilibrium* if for every player $i \in [n]$ and every action $a_i \in A_i$ we have $u_i(x_i, x_{-i}) \geq u_i(a_i, x_{-i}) - \epsilon$. For $\epsilon = 0$ such a profile is called a *Nash equilibrium.*

3.2 Optimization Definitions

We consider the total search problem GD-FixedPoint that naturally arise in the context of continuous optimization via gradient-based methods:

[18]The converse is true (i.e., a congestion game is a c-polytensor identical interest game) whenever for every facility the number of players that can potentially use it (i.e., have an action that includes this facility) is bounded by c.

Meta Definition 1 (GD-FixedPoint)**.**

Given a smooth function $\phi : [0,1]^J \to [0,1]$ and an $\epsilon > 0$ find a point $x \in [0,1]^J$ such that

(1) $\nabla_j \phi(x) \geq -\epsilon$ for every dimension $j \in J$ such that $x_j > 0$.
(2) $\nabla_j \phi(x) \leq \epsilon$ for every dimension $j \in J$ such that $x_j < 1$.

Namely we are searching for a point x such that a movement in the gradient direction will either be impossible (i.e., when $x_j = 0$ the j-th coordinate cannot be decreased and when $x_j = 1$ the j-th coordinate cannot be increased) or alternatively, there will be no improvement (approximately zero gradient).

So far we neither specified the form in which the function ϕ is given to the algorithm, nor provided any special properties of ϕ. As we will see, variants of GD-FixedPoint capture the class CCLS and the problem KKT[19] that have been defined in [27].

In the first, most general, variant of GD-FixedPoint ϕ is given by an arbitrary well-behaved arithmetic circuit with $\{+, -, \times, \times\zeta, \max, \min, >\}$ gates, where $\times\zeta$ denotes multiplication by a constant. max, min gates are not smooth and $>$ gate is not even continuous, but it is promised[20] that $\nabla\phi$ is defined and α-Lipschitz in the $\ell_2 - \ell_2$ sense (the parameter α is given as an input to the algorithm). The *well-behaved* desideratum restricts that the number $\times$ gates on any input-output path is at most logarithmic (in the size of the circuit). This desideratum is important to avoid doubly-exponential numbers from repeated squaring. See [35] for a detailed discussion.

[19][35] also consider a problem that they call KKT which corresponds to the most general variant of GD-FixedPoint. They leave as an open problem the computational complexity of [27]'s (explicit) KKT, which we settle in this paper.

[20]To make the problem total, the algorithm may either return a solution or a certificate that this promise is violated; however the PPAD ∩ PLS-complete instances [35] always satisfy the promise, so it suffices to prove hardness for this special case.

In particular, we focus on a special case of this variant where the domain of the potential is $[0,1]^2$ (i.e., $|J| = 2$). We call this problem 2D-GD-FixedPoint. By [35], this special case is as hard as the general high dimensional case:

Theorem ([35]). *2D-GD-FixedPoint is* PPAD $\cap$ PLS*-complete.*

The second variant of GD-FixedPoint is Con-GD-FixedPoint where in addition to the α-smoothness it deals with *componentwise concave* potentials. Namely, for every $j \in J$ and every fixed $x^*_{-j} \in [0,1]^{J \setminus \{j\}}$ the function $\phi(x_j, x^*_{-j}) : [0,1] \to [0,1]$ should be concave. To guarantee a total variant of the problem we allow the algorithm to output either a gradient fixed point or an evidence for componentwise concavity violation that comes in the form of a triplet of points. We will see (Lemma 1) that this problem is equivalent to CCLS.

The third variant of GD-FixedPoint is Expl-GD-FixedPoint considers instances where ϕ is a polynomial. Here the input is given by an explicitly description of monomials' coefficients. Note that this variant is a natural problem since ϕ is not described by a circuit. We will see (Lemma 2) that this problem is equivalent to KKT.

Finally the last, most restrictive, variant of GD-FixedPoint is c-Deg-GD-FixedPoint where ϕ is given explicitly and a sum of constant degree c monomials, and each monomial is componentwise concave. This problem is a special case of Con-GD-FixedPoint since each monomial is componentwise concave (and so is the sum) and it is also a special case of Expl-GD-FixedPoint because monomials are given explicitly (and their total number does not exceed $|J|^c$).

3.2.1 Relation to Other Problems in the Literature. Now we relate the defined above variants of GD-FixedPoint with the existing continuous optimization problems that have been defined in [27].

The problem CCLS that defines the class CCLS (see [27]) is defined as follows:

Definition 1 (CCLS [27])**.**

Input: A potential function $\phi : [0,1]^J \to [0,1]$ that is given an arithmetic circuit and two constants $\bar{\epsilon}$ and δ.

Output: Any of the following:

- a point $x \in [0,1]^J$ such that $\phi(x) \geq \phi((1-\delta)x_j, x_{-j}) - \bar{\epsilon}$ and $\phi(x) \geq \phi((1-\delta)x_j + \delta, x_{-j}) - \bar{\epsilon}$ for every $j \in J$;
- a pair of points that exhibit a violation of Lipschitz condition; or
- a triplet of points that exhibit a violation of componentwise concavity violation.

The two terms $(1-\delta)x_j$ and $(1-\delta)x_j + \delta$ follow from the multiplicative deviation variant of CCLS $x_j \to (1-\delta)x_j + \delta e_j$.

Remark 4. Our definition of the CCLS problem is slightly more restrictive than the original definition in [27]: (i) we require that the domain is a hypercube rather than the more general product of simplices; and (ii) we require that the function has bounded smoothness. Since we show that our variant is also PLS $\cap$ PPAD-complete, it implies in particular that the definitions are equivalent.

The equivalence of the CCLS problem and our formulation via the GD-FixedPoint is stated in the following lemma.

Lemma 1. *CCLS is computationally equivalent to* Con-GD-FixedPoint.

Proof. For the reductions in both direction if the output is a triplet of points violating concavity this output is valid for the reduced problem as well. So we restrict attention to the interesting case where the output is a gradient fixed point (in one direction of the reduction) or a solution for CCLS (in the opposite direction reduction).

Given an instance of CCLS $(\phi, \bar{\epsilon}, \delta)$ we set[21] $\epsilon = \bar{\epsilon}$. Let x be a GD-FixedPoint-Con solution for (ϕ, ϵ). By concavity we have

$$\begin{aligned}\phi((1-\delta)x_j, x_{-j}) &\leq \phi(x) - \delta x_j \nabla_j \phi(x) \\ &\leq \phi(x) + \delta x_j \epsilon \\ &\leq \phi(x) + \epsilon\delta \\ &\leq \phi(x) + \bar{\epsilon},\end{aligned}$$

and similarly also

$$\begin{aligned}\phi((1-\delta)x_j + \delta, x_{-j}) &\leq \phi(x) + \delta(1-x_j)\nabla_j \phi(x) \\ &\leq \phi(x) + \delta(1-x_j)\epsilon \\ &\leq \phi(x) + \epsilon\delta \\ &\leq \phi(x) + \bar{\epsilon}.\end{aligned}$$

Conversely, given an instance (ϕ, ϵ) of GD-FixedPoint-Expl we set $\bar{\epsilon} = \frac{\epsilon^3}{64J\alpha^2}$ and $\delta = \frac{\epsilon}{4\alpha}$. Let $x \in [0,1]^J$ be the CCLS solution of $(\phi, \bar{\epsilon}, \delta)$. We δ-round the point x to the boundaries $\{0,1\}$ as follows. For every $j \in J$ we set

$$\bar{x}_j = \begin{cases} 0 & \text{if } x_j \in [0, \delta) \\ x_j & \text{if } x_j \in [\delta, 1-\delta] \\ 1 & \text{if } x_j \in (1-\delta, 1] \end{cases}$$

We argue that $\bar{x}$ is a solution for GD-FixedPoint. For every dimension $j \in J$ we consider three cases.

Case 1: $x_j \in [0, \delta)$. In such a case $\bar{x}_j = 0$ and hence we only need to prove that $\nabla_j \phi(\bar{x}) \leq \epsilon$. Since x and $\bar{x}$ are $(\bar{\epsilon}J)$-close, by the α-Lipschitzness of the gradient we have $|\nabla_j(x) - \nabla_j(\bar{x})| \leq \bar{\epsilon}J\alpha \leq \frac{\epsilon}{2}$. Therefore, it is sufficient to prove that $\nabla_j \phi(x) \leq \frac{\epsilon}{2}$.

We know that $\phi((1-\delta)x_j + \delta, x_{-j}) - \phi(x) \leq \bar{\epsilon}$. By the mean value Theorem there exists $0 \leq \epsilon' \leq \delta(1-x_j)$ such that

$$\nabla_j \phi(x + \epsilon' e_j) = \frac{\phi((1-\delta)x_j + \delta, x_{-j}) - \phi(x)}{\delta(1-x_j)} \leq \frac{\bar{\epsilon}}{\delta(1-x_j)}.$$

By the Lipschitzness of the gradient we deduce that $\nabla_j \phi(x) \leq \frac{\bar{\epsilon}}{\delta(1-x_j)} + \delta x_j \alpha \leq \frac{\bar{\epsilon}}{\delta^2} + \delta\alpha \leq \frac{\epsilon}{4} + \frac{\epsilon}{4}$.

Case 2: $x_j \in (1-\delta, 1]$. We apply similar arguments to those of Case 1, but this time we consider the inequality $\phi(x) - \phi((1-\delta)x_j, x_{-j}) \geq \bar{\epsilon}$.

Case 3: $x_j \in [\delta, 1-\delta]$. We consider both inequalities $\phi((1-\delta)x_j + \delta, x_{-j}) - \phi(x) \leq \bar{\epsilon}$ and $\phi(x) - \phi((1-\delta)x_j, x_{-j}) \geq \bar{\epsilon}$ and apply the arguments of Cases 1 and 2. □

The KKT problem that has been defined in [27] is inspired by the following (multidimensional) formulation of Taylor's Theorem:

[21] For this direction of the reduction the parameter δ turns out to be irrelevant.

Taylor's Theorem (See [27] Lemma 3.1). *For every function $\phi : \mathbb{R}^J \to \mathbb{R}$ with α-Lipschitz gradients $\nabla\phi$ in the $\ell_2 - \ell_2$ sense and for every $x_0, x \in \mathbb{R}^J$ we have*

$$|\phi(x) - \phi(x_0) - \nabla\phi(x_0)\cdot(x - x_0)| \leq \frac{\alpha}{2}\|x - x_0\|_2^2.$$

The problem KKT is defined as follows:

Definition 2 (KKT [27]).

Input: A potential function $\phi : [0,1]^J \to [0,1]$ that is given by coefficients of monomials in $|J|$ variables[22], $\bar{\epsilon}$ and κ.

Output: A point $x \in [0,1]^J$ such that $\phi(x) \geq \phi(y) - \frac{\alpha}{2}\bar{\epsilon}^2 - \kappa$ for every $y \in B(x,\bar{\epsilon}) \cap [0,1]^J$, when $B(x,\bar{\epsilon})$ is the ℓ_2-ball of radius $\bar{\epsilon}$ around x.

The following lemma shows the equivalence of the KKT problem and the defined above GD-FixedPoint-Expl variant of gradient-dynamic fixed point problem.

Lemma 2. KKT *is computationally equivalent to* Expl-GD-FixedPoint.

Proof. Given an instance of KKT $(\phi,\bar{\epsilon},\kappa)$ we set $\epsilon = \frac{\kappa}{J\bar{\epsilon}}$ and we apply the GD-FixedPoint-Expl algorithm for (ϕ,ϵ). By Taylor's Theorem for every $y \in B(x,\bar{\epsilon})$ we have

$$\phi(x) \geq \phi(y) - \nabla\phi\cdot(y-x) - \frac{\alpha}{2}\bar{\epsilon}^2 \geq \phi(y) - \frac{\alpha}{2}\bar{\epsilon}^2 - \epsilon J\bar{\epsilon} \geq \phi(y) - \frac{\alpha}{2}\bar{\epsilon}^2 - \kappa.$$

Conversely, given an instance (ϕ,ϵ) of GD-FixedPoint-Expl we set $\bar{\epsilon} = \frac{\epsilon}{6J\alpha}$ and $\kappa = \frac{\epsilon^2}{24J\alpha}$. Let $x \in [0,1]^J$ be the KKT solution of $(\phi,\bar{\epsilon},\kappa)$. We $\bar{\epsilon}$-round the point x to the boundaries $\{0,1\}$ as follows. For every $j \in J$ we set

$$\overline{x}_j = \begin{cases} 0 & \text{if } x_j \in [0,\bar{\epsilon}) \\ x_j & \text{if } x_j \in [\bar{\epsilon}, 1-\bar{\epsilon}] \\ 1 & \text{if } x_j \in (1-\bar{\epsilon}, 1] \end{cases}$$

We argue that $\overline{x}$ is a solution for GD-FixedPoint. For every dimension $j \in J$ we consider three cases.

Case 1: $x_j \in [0,\bar{\epsilon})$. In such a case $\overline{x}_j = 0$ and hence we only need to prove that $\nabla_j\phi(\overline{x}) \leq \epsilon$. Since x and $\overline{x}$ are $(\bar{\epsilon}J)$-close, by the α-Lipschitzness of the gradient we have $|\nabla_j(x) - \nabla_j(\overline{x})| \leq \bar{\epsilon}J\alpha \leq \frac{\epsilon}{2}$. Therefore, it is sufficient to prove that $\nabla_j\phi(x) \leq \frac{\epsilon}{2}$.

Consider a point $y = x + \bar{\epsilon}e_j$ which necessarily belongs to the hypercube. By the KKT condition we know that $\phi(y) - \phi(x) \leq \frac{\alpha}{2}\bar{\epsilon}^2 + \kappa$ and hence $\frac{\phi(y)-\phi(x)}{\bar{\epsilon}} \leq \frac{\alpha}{2}\bar{\epsilon} + \frac{\kappa}{\bar{\epsilon}}$. By the mean value Theorem there exists $0 \leq \epsilon' \leq \bar{\epsilon}$ such that $\nabla_j\phi(x + \epsilon' e_j) \leq \frac{\alpha}{2}\bar{\epsilon} + \frac{\kappa}{\bar{\epsilon}}$. By the Lipschitzness of the gradient we get $\nabla_j\phi(x) \leq \frac{3\alpha}{2}\bar{\epsilon} + \frac{\kappa}{\bar{\epsilon}} \leq \frac{\epsilon}{4} + \frac{\epsilon}{4}$.

Case 2: $x_j \in (1-\bar{\epsilon}, 1]$. We consider the point $y = x - \bar{\epsilon}e_j$ and apply the symmetric arguments to those of Case 1.

Case 3: $x_j \in [\bar{\epsilon}, 1-\bar{\epsilon}]$. We consider both points $y = x \pm \bar{\epsilon}e_j$ and apply the arguments of Cases 1 and 2.

□

[22] We follow the convention of [27] and restrict the KKT problem to explicit polynomials. To avoid confusions, the Karush–Kuhn–Tucker (KKT) conditions are valid for arbitrary potential functions.

3.3 Formal Statement of Our Results

Our main result is as follows:

Theorem 1. *The problems* Congestion, 5-Polytensor-IdenticalInterest, Con-GD-FixedPoint, KKT, *and* Deg-5-GD-FixedPoint *are* PPAD ∩ PLS*-complete.*

As a corollary, we obtain the following equivalence of complexity classes.

Corollary 2. CCLS = PPAD ∩ PLS.

3.4 Organization

Our main technical contribution is the reduction

$$\text{2D-GD-FixedPoint} \leq \text{5-Polytensor-IdenticalInterest}. \qquad (2)$$

In Section 4 we provide intuition and ideas of the main reduction. The actual proof appears in the full version.

(2) implies that 5-Polytensor-IdenticalInterest and Congestion are PPAD ∩ PLS-complete. The only non-obvious aspect of extending this hardness to variants of GD-FixedPoint, is that 5-Polytensor-IdenticalInterest naturally corresponds to a potential function over a product of simplices (where each simplex corresponds to the feasible mixed strategies of each player), whereas GD-FixedPoint is defined with hypercube domain. In the full version we show how to handle non-hypercube domains.

4 MAIN RESULT: AN INFORMAL TECHNICAL OVERVIEW

In this section we provide a more detailed overview of our reduction, complementing the highlights described in Section 2.

4.1 The Utility

We find it useful for the reader to jump directly to the formula of the game's utility, explain the notations in the utility, and explain which role each utility term plays in the reduction. See full version for details. We describe the game in the form of a sum of *all* local interactions, which is equivalent (with respect to Nash equilibria analysis) to a polytensor identical-interest game; see Remark 2.

The parameters that appear in the utility satisfy $\epsilon_C \gg \epsilon_P \gg \epsilon_G \gg \epsilon_T$.

Imitation utility. The index $p \in P$ denotes a point in the sampling. The index $j \in J$ denotes a coordinate. The action of the guide player is denoted by g. The term $\overline{x}_j^p$ denotes the j-th coordinate of the point p (recall that this coordinate, as any other coordinate in the game, is chosen by K bit players and one veto player). The term $\overline{y}_j$ denotes the j-th coordinate of the point p. The term $i(p,j)\epsilon_S$ is the sampling shift of y which the x tries to imitate.

The quadratic imitation loss plays a central role in our reduction. This is a convenient imitation loss to work with due to two reasons. First, in a mixed Nash equilibrium analysis, even if both the x_j^p team and the y_j team are playing mixed strategies, we have a clean formula for the expected utility from which it is clear that the x team tries to imitate the expectation of $\overline{y}_j$. Second, the fact that it is a finite degree polynomial allows us to argue that this term is in fact c-uniform polymatrix game. Note that the imitation is the most significant term in the utility and it remains most significant even if

$$
\begin{aligned}
u = &- \sum_{p\in P, j\in J} (1 + \mathbf{1}_{g=p})[\overline{x}_j^p - \overline{y}_j - i(p,j)\epsilon_S]^2 && \text{(imitation)}\\
&- \epsilon_C \sum_{p\in P, l\in\mathcal{L}} (\epsilon_T + \mathbf{1}_{[g=p]})2^{-2d_l} w_l^p && \text{(circuit)}\\
&+ \epsilon_P \sum_{p\in P} \mathbf{1}_{[g=p]}(\phi^p - c^p) && \text{(potential)}\\
&+ \epsilon_G \sum_{p\in P, j\in J} \overline{y}_j \mathbf{1}_{[g=p]} \triangle_j^p && \text{(gradient)}\\
&- \epsilon_T \sum_{p\in P, j\in J, k\in[K]} [\tilde{x}_{j,k}^p - \overline{y}_j - i(p,j)\epsilon_S]^2 && \text{(x-bit imitation)}\\
&- \epsilon_T \sum_{p\in P, j\in J, k\in[K]} (1 + \mathbf{1}_{g=p})[\overline{x}_j^p - \tilde{y}_{j,k} - i(p,j)\epsilon_S]^2 && \text{(y-bit imitation)}\\
&- \epsilon_T\epsilon_G \sum_{p\in P, j\in J, k\in[K]} \tilde{y}_{j,k}\mathbf{1}_{[g=p]}\triangle_j^p && \text{(bit gradient)}\\
&+ \epsilon_T \sum_{r\in R} m_r^x + \epsilon_T \sum_{j\in J} m_j^y && \text{(veto).}
\end{aligned}
$$

Figure 2: The utility

the guide player does not choose the point p (i.e., even if $\mathbf{1}_{[g=p]} = 0$). Having said that, note that in case the imitation is almost perfect, for instance if $\overline{x}_j^p$ is pure and is located $\epsilon \approx 2^{-K}$ close to its target $\mathbb{E}[\overline{y}_j + i(p,j)\epsilon_S]$ the gain from making the imitation perfect improves the imitation loss only by $O(2^{-2K})$ which is negligible relative to the circuit, the potential, and the gradient utility terms.

Circuit utility. The index $l \in \mathcal{L}$ refers to a line (wire) in a circuit. Note that every point has its own circuit to compute ϕ and $(\triangle_j)_{j\in J}$. The term d_l is the depth of the gadget. The term w_l^p denotes the indicator of whether the binary action played by the player that is located at l-th line in the p-th circuit is *wrong*; namely, it is not the correct output of the corresponding gate given the input players' actions.

By the exponential decay of the weights of wrong computations, the incentive of a gadget to match her predecessors is 4 times larger than her gain from correct computations of her successors. This creates an incentive for all gadget players to perform correct computations in case the input to the circuit is pure.[23]

Note that in case the guide player chooses a point, its weight for correct computations is relatively high (ϵ_C); whereas in case the guide player chooses a point with probability 0, the incentive for correct computation is tiny (ϵ_T). This allows points that are located close to the boundary and are not chosen by the guide player to pass though the boundary because for such points imitation is more significant. The tiny incentive ϵ_T for correct computations appears there to reach updating of the circuit to the new input after x_j^p has passed the boundary.

Potential utility. The term ϕ^p denotes the calculated potential by the circuit of the point p. Note that only potentials that are chosen by the guide player with positive probability appear in the utility. We prove that the guide player does not choose the problematic points with mixed input that are located on the $2^{-N_{in}}$ grid. Therefore, the computed potential in the utility is in fact meaningful correct potential rather than some unexpected potential that is a result of mixture of the predecessors.

Gradient utility. The term $\triangle^p$ denotes the gradient[24] estimated by the circuit of the point p. Similarly to the potential utility, we note that only the values of circuits that are chosen by the guide player with positive probability are counted. The factor of $\overline{y}_j$ incentivizes the y-team to increase (decrease) $\overline{y}_j$ if $\triangle_j > 0$ ($\triangle_j < 0$).

x-bit imitation utility. The index $k \in [K]$ refers to the binary bits that create the real numbers $\overline{x}_j^p$. The term $\tilde{x}_{j,k}^p$ denotes the number $\overline{x}_j^p$ when we replace the k-th bit to be the one that the k-th player choose (even if she was vetoed). The role of this term is to provide an incentive for vetoed bits to update their action to the one that veto player choose for them (recall that when a bit player is vetoed, her decision does not affect $\overline{x}_j^p$).

y-bit imitation utility. Similar to x-bit imitation.

Bit gradient utility. This is a technical term that allows us to apply on the y-team similar arguments to those we apply for the x^p teams. Note that the weight of this term is extremely low: $\epsilon_T\epsilon_G$.

Veto utility. The index $r \in R$ refers to real numbers chosen by the x-teams (the real numbers that generate the points x^p). The integer number $m_r^x \in [K+1]$ is chosen by the veto player of the x_r team and it denotes the bit starting from which the veto player implies her veto ($m_r^x = K+1$ means no veto). This utility incentivizes the veto player to impose veto on less bits. This utility term is needed

[23]It remains unclear what the circuit players will do in case where the input bits are mixed.

[24]It is actually more convenient to calculate $\triangle_j(x) = \phi(x + 2^{-N_{in}}) - \phi(x)$ rather than the gradient whose approximation is $\frac{\phi(x+2^{-N_{in}})-\phi(x)}{2^{-N_{in}}}$. Namely, $\triangle_j \approx 2^{-N_{in}}\nabla_j$.

in order that after the bit players match their bits with the veto player, the veto on them would be canceled.

4.2 A Better-Reply Path

Another angle that provides intuition about the reduction is a description of a better-reply path that converges to a gradient decent fixed point in our game. Roughly speaking the x and the y teams almost continuously move in the direction of the gradient until they reach a gradient decent fixed point. (The move is continuous up to a very small step size $\approx 2^{-K}$.) Below we describe the steps of this process. For simplicity we focus on a single coordinate; for multiple coordinates these steps simply happen simultaneously.

(0) In this discussion we consider the case where the initial state of the best-reply dynamic is already quite arranged[25]: the x-team and the y-team players are playing pure; all the x-teams perfectly imitate the corresponding y shifts; no veto is imposed by the veto player at any team; all the x-teams and the y team are located far from the $2^{-N_{in}}$-grid and we denote $x^p \in [a2^{-N_{in}}, (a+1)2^{-N_{in}})$ for some integer $a \in [2^{N_{in}}]$; and finally we assume that in this region $\nabla\phi(a2^{-N_{in}}) > 0$.

(1) Any circuit player that is located on the input gadgets observes a bit player who plays pure and whom she tries to imitate. Therefore, this input circuit player strictly prefers to match her bit, so the inputs to the circuit is $a2^{-N_{in}}$ for every circuit. We proceed inductively with the depth of the circuit to deduce that all the circuits correctly compute $\phi(a2^{-N_{in}})$ and correctly compute $\nabla(a2^{-N_{in}})$.

(2) The guide player sees identical samples in terms of the imitation loss and in terms of the computed potential and might play mix at this point.

(3) The y-team will be better off by deviating to $y + 2^{-K}$: this causes an imitation loss of $|P|2^{-2K}$ (2^{-2K} for each one of the $|P|$ x-teams) but increases the gradient utility by $\epsilon_G 2^{-K}\nabla(a2^{-N_{in}}) \gg |P|2^{-2K}$. If the least significant bit of y is 0 it is simply flipped to 1; otherwise consider the suffix 01...1 beginning with the least significant 0: the y-vetor player prefers to impose a 10...0 veto over the existing 01...1.

(4) After the bit players corresponding to the suffix of y are vetoed, their action affects only the y-bit imitation term. In this term their bit is aggregated with the veto player's assignments for the other bits. Therefore, every vetoed bit player prefers to match the action of the veto player.

(5) After all vetoed bit players have updated their bits to 100...0 the veto player prefers to cancel her veto on these players because of the veto utility.

(6) After the y team has moved by 2^{-K} the x-teams prefer to move by 2^{-K} too in order to perfectly imitate their shifts of the y-team. Note that such a move does not affect the first N_{in} bits and hence does not affect the circuit utility which continues to be perfect. Namely, each x-team separately updates the actions according to Steps (3),(4), and (5).

(7) The x and y team proceed moving in the positive direction by repeatedly applying Steps (3)-(6) until one of these teams reaches the boundary $(a+1)2^{-N_{in}}$.

(8) Let x_{-1}, x_0, x_1 denote three x-samples, trying to imitate y with negative, zero, and positive shift, respectively. The first team that will reach the boundary is the x_1 team. At this point it will stop increasing x_1 because such a change will cause wrong computations of the circuit. Meanwhile teams x_0 and x_{-1} proceed updating their actions and proceed imitating their shifts of y perfectly.

(9) Now the guide player is no longer indifferent between all samples and she prefers to avoid choosing x_1 because this sample has non-perfect imitation while the computed potential at x_1 is the same as in x_0 and in x_{-1}.

(10) Once the guide player does not choose x_1 the weight of the circuit in this team becomes negligible and then the x_1-team prefers to increase the value x_1. In particular the N_{in} most significant bits become $(a+1)2^{-N_{in}}$.

(11) The circuit players of the x_1 team update their actions according to Step (2).

(12) The computed potential $\phi((a+1)2^{-N_{in}})$ at x_1 team is now *higher* than the computed potential $\phi(a2^{-N_{in}})$ at x_0 and x_{-1} teams and hence the guide player switches to choose team x_1.

(13) All x-teams and the y-team proceed to increase their values by 2^{-K} by repeatedly applying Steps (3)-(6) until both other teams x_0 and x_{-1} also pass the boundary from $a2^{-N_{in}}$ to $(a+1)2^{-N_{in}}$. Now we got back to the initial state at Step (0) but now all the x-teams and the y-team are located in the next grid cell $[(a+1)2^{-N_{in}}, (a+2)2^{-N_{in}})$.

(14) Players repeatedly apply Steps (1)-(13) until they reach either the boundary (0 or 1) or they reach a point x with approximately equal potential values at $a2^{-N_{in}}$ and at $(a+1)2^{-N_{in}}$. In the former case the procedure terminates are a boundary point 0 (1) with negative (positive) gradient. In the latter case the procedure terminates at an interior point with approximately zero gradient.

[25]The assumption that the initial state of the dynamic is arranged substantially simplifies the arguments. For an arbitrary mixed action profile it is more involved to identify the player that has an incentive to deviate. These cases are treated in the formal proof

REFERENCES

[1] Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. 2008. On the impact of combinatorial structure on congestion games. *J. ACM* 55, 6 (2008), 25:1–25:22. https://doi.org/10.1145/1455248.1455249

[2] Heiner Ackermann and Alexander Skopalik. 2008. Complexity of Pure Nash Equilibria in Player-Specific Network Congestion Games. *Internet Mathematics* 5, 4 (2008), 323–342. https://doi.org/10.1080/15427951.2008.10129170

[3] Sebastian Aland, Dominic Dumrauf, Martin Gairing, Burkhard Monien, and Florian Schoppmann. 2011. Exact Price of Anarchy for Polynomial Congestion Games. *SIAM J. Comput.* 40, 5 (2011), 1211–1233. https://doi.org/10.1137/090748986

[4] Omer Angel, Sebastien Bubeck, Yuval Peres, and Fan Wei. 2017. Local max-cut in smoothed polynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, Hamed Hatami, Pierre McKenzie, and Valerie King (Eds.). ACM, 429–437. https://doi.org/10.1145/3055399.3055402

[5] Baruch Awerbuch, Yossi Azar, and Amir Epstein. 2013. The Price of Routing Unsplittable Flow. *SIAM J. Comput.* 42, 1 (2013), 160–177. https://doi.org/10.1137/070702370

[6] Yakov Babichenko. 2016. Query complexity of approximate Nash equilibria. *Journal of the ACM (JACM)* 63, 4 (2016), 36.

[7] Yakov Babichenko and Aviad Rubinstein. 2017. Communication complexity of approximate Nash equilibria. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 878–889.

[8] Yakov Babichenko and Aviad Rubinstein. 2020. Communication complexity of Nash equilibrium in potential games. In *FOCS 2020*.

[9] Ali Bibak, Charles Carlson, and Karthekeyan Chandrasekaran. 2019. Improving the smoothed complexity of FLIP for max cut problems. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, Timothy M. Chan (Ed.). SIAM, 897–916. https://doi.org/10.1137/1.9781611975482.55

[10] Avrim Blum, MohammadTaghi Hajiaghayi, Katrina Ligett, and Aaron Roth. 2008. Regret minimization and the price of total anarchy. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, Cynthia Dwork (Ed.). ACM, 373–382. https://doi.org/10.1145/1374376.1374430

[11] Shant Boodaghians, Rucha Kulkarni, and Ruta Mehta. 2020. Smoothed Efficient Algorithms and Reductions for Network Coordination Games. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA (LIPIcs, Vol. 151)*, Thomas Vidick (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 73:1–73:15. https://doi.org/10.4230/LIPIcs.ITCS.2020.73

[12] Sébastien Bubeck. 2015. Convex Optimization: Algorithms and Complexity. *Found. Trends Mach. Learn.* 8, 3-4 (2015), 231–357. https://doi.org/10.1561/2200000050

[13] Sébastien Bubeck and Dan Mikulincer. 2020. How to Trap a Gradient Flow. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria] (Proceedings of Machine Learning Research, Vol. 125)*, Jacob D. Abernethy and Shivani Agarwal (Eds.). PMLR, 940–960. http://proceedings.mlr.press/v125/bubeck20b.html

[14] Josh Buresh-Oppenheim and Tsuyoshi Morioka. 2004. Relativized NP Search Problems and Propositional Proof Systems. In *19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA*. IEEE Computer Society, 54–67. https://doi.org/10.1109/CCC.2004.1313795

[15] Samuel R. Buss and Alan S. Johnson. 2012. Propositional proofs and reductions between NP search problems. *Ann. Pure Appl. Log.* 163, 9 (2012), 1163–1182. https://doi.org/10.1016/j.apal.2012.01.015

[16] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. 2018. Accelerated Methods for NonConvex Optimization. *SIAM J. Optim.* 28, 2 (2018), 1751–1772. https://doi.org/10.1137/17M1114296

[17] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. 2020. Lower bounds for finding stationary points I. *Math. Program.* 184, 1 (2020), 71–120. https://doi.org/10.1007/s10107-019-01406-y

[18] Xi Chen, Decheng Dai, Ye Du, and Shang-Hua Teng. 2010. On the complexity of equilibria in markets with additively separable utilities. In *Proceedings of the Behavioral and Quantitative Game Theory - Conference on Future Directions, BQGT '10, Newport Beach, California, USA, May 14-16, 2010*, Moshe Dror and Greys Sosic (Eds.). ACM, 61:1. https://doi.org/10.1145/1807406.1807467

[19] Xi Chen and Xiaotie Deng. 2009. On the complexity of 2D discrete fixed point problem. *Theor. Comput. Sci.* 410, 44 (2009), 4448–4456. https://doi.org/10.1016/j.tcs.2009.07.052

[20] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. 2009. Settling the complexity of computing two-player Nash equilibria. *J. ACM* 56, 3 (2009), 14:1–14:57. https://doi.org/10.1145/1516512.1516516

[21] Xi Chen, David Durfee, and Anthi Orfanou. 2015. On the Complexity of Nash Equilibria in Anonymous Games. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 381–390. https://doi.org/10.1145/2746539.2746571

[22] Xi Chen, Chenghao Guo, Emmanouil-Vasileios Vlatakis-Gkaragkounis, Mihalis Yannakakis, and Xinzhi Zhang. 2020. Smoothed complexity of local max-cut and binary max-CSP. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy (Eds.). ACM, 1052–1065. https://doi.org/10.1145/3357713.3384325

[23] Xi Chen, Dimitris Paparas, and Mihalis Yannakakis. 2017. The Complexity of Non-Monotone Markets. *J. ACM* 64, 3 (2017), 20:1–20:56. https://doi.org/10.1145/3064810

[24] Artur Czumaj and Berthold Vöcking. 2007. Tight bounds for worst-case equilibria. *ACM Trans. Algorithms* 3, 1 (2007), 4:1–4:17. https://doi.org/10.1145/1219944.1219949

[25] Constantinos Daskalakis. 2008. *The Complexity of Nash Equilibria.* Ph.D. Dissertation. University of California, Berkeley.

[26] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. 2009. The complexity of computing a Nash equilibrium. *SIAM J. Comput.* 39, 1 (2009), 195–259.

[27] Constantinos Daskalakis and Christos Papadimitriou. 2011. Continuous local search. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 790–804.

[28] Constantinos Daskalakis and Christos H. Papadimitriou. 2020. Continuous Local Search - Corrigendum. http://people.csail.mit.edu/costis/CLS-corrigendum.pdf

[29] Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis. 2020. The Complexity of Constrained Min-Max Optimization. *CoRR* abs/2009.09623 (2020). arXiv:2009.09623 https://arxiv.org/abs/2009.09623

[30] Constantinos Daskalakis, Christos Tzamos, and Manolis Zampetakis. 2018. A converse to Banach's fixed point theorem and its CLS-completeness. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 44–50.

[31] Argyrios Deligkas, John Fearnley, and Rahul Savani. 2020. Tree Polymatrix Games are PPAD-hard. *CoRR* abs/2002.12119 (2020). arXiv:2002.12119 https://arxiv.org/abs/2002.12119

[32] Kousha Etessami, Christos H. Papadimitriou, Aviad Rubinstein, and Mihalis Yannakakis. 2020. Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*. 18:1–18:19. https://doi.org/10.4230/LIPIcs.ITCS.2020.18

[33] Michael Etscheid and Heiko Röglin. 2017. Smoothed Analysis of Local Search for the Maximum-Cut Problem. *ACM Trans. Algorithms* 13, 2 (2017), 25:1–25:12. https://doi.org/10.1145/3011870

[34] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. 2004. The complexity of pure Nash equilibria. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM, 604–612.

[35] John Fearnley, Paul W. Goldberg, Alexandros Hollender, and Rahul Savani. 2021. The Complexity of Gradient Descent: CLS = PPAD ∩ PLS. In *STOC 2021*.

[36] John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. 2017. CLS: New Problems and Completeness. *CoRR* abs/1702.06017 (2017). arXiv:1702.06017 http://arxiv.org/abs/1702.06017

[37] John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. 2020. Unique end of potential line. *J. Comput. Syst. Sci.* 114 (2020), 1–35. https://doi.org/10.1016/j.jcss.2020.05.007

[38] John Fearnley and Rahul Savani. 2020. A faster algorithm for finding Tarski fixed points. *CoRR* abs/2010.02618 (2020). arXiv:2010.02618 https://arxiv.org/abs/2010.02618

[39] Aris Filos-Ratsikas and Paul W. Goldberg. 2018. Consensus halving is PPA-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, Ilias Diakonikolas, David Kempe, and Monika Henzinger (Eds.). ACM, 51–64. https://doi.org/10.1145/3188745.3188880

[40] Anat Ganor, Karthik C. S., and Dömötör Pálvölgyi. 2019. On Communication Complexity of Fixed Point Computation. *CoRR* abs/1909.10958 (2019). arXiv:1909.10958 http://arxiv.org/abs/1909.10958

[41] Jugal Garg and Vijay V. Vazirani. 2014. On Computability of Equilibria in Markets with Production. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, Chandra Chekuri (Ed.). SIAM, 1329–1340. https://doi.org/10.1137/1.9781611973402.98

[42] Bernd Gärtner, Thomas Dueholm Hansen, Pavel Hubáček, Karel Král, Hagar Mosaad, and Veronika Slívová. 2018. ARRIVAL: Next Stop in CLS. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*. 60:1–60:13. https://doi.org/10.4230/LIPIcs.ICALP.2018.60

[43] Elazar Goldenberg and Karthik C. S. 2020. Hardness Amplification of Optimization Problems. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA (LIPIcs, Vol. 151)*, Thomas Vidick (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 1:1–1:13. https://doi.org/10.4230/LIPIcs.ITCS.2020.1

[44] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. 2019. Adventures in Monotone Complexity and TFNP. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*. 38:1–38:19. https://doi.org/10.4230/LIPIcs.ITCS.2019.38

[45] Mika Göös and Aviad Rubinstein. 2018. Near-Optimal Communication Lower Bounds for Approximate Nash Equilibria. In *59th IEEE Annual Symposium on Foundations of Computer Science, (FOCS) 2018, Paris, France, October 7-9, 2018*. 397–403. https://doi.org/10.1109/FOCS.2018.00045

[46] Tobias Harks, Martin Hoefer, Max Klimm, and Alexander Skopalik. 2013. Computing pure Nash and strong equilibria in bottleneck congestion games. *Math. Program.* 141, 1-2 (2013), 193–215. https://doi.org/10.1007/s10107-012-0521-3

[47] Joseph T Howson Jr. 1972. Equilibria of polymatrix games. *Management Science* 18, 5-part-1 (1972), 312–318.

[48] David S Johnson, Christos H Papadimitriou, and Mihalis Yannakakis. 1988. How easy is local search? *Journal of computer and system sciences* 37, 1 (1988), 79–100.

[49] Elias Koutsoupias and Christos H. Papadimitriou. 2009. Worst-case equilibria. *Comput. Sci. Rev.* 3, 2 (2009), 65–69. https://doi.org/10.1016/j.cosrev.2009.04.003

[50] Mark W. Krentel. 1990. On Finding and Verifying Locally Optimal Solutions. *SIAM J. Comput.* 19, 4 (1990), 742–749. https://doi.org/10.1137/0219052

[51] Ruta Mehta. 2014. Constant rank bimatrix games are PPAD-hard. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*. 545–554. https://doi.org/10.1145/2591796.2591835

[52] Dov Monderer and Lloyd S Shapley. 1996. Potential games. *Games and Economic Behavior* 14, 1 (1996), 124–143.

[53] Aniket Murhekar and Ruta Mehta. 2020. Approximate Nash Equilibria of Imitation Games: Algorithms and Complexity. In *Proceedings of the 19th International*

Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020, Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, 887–894. https://dl.acm.org/doi/abs/10.5555/3398761.3398865

[54] Christos H Papadimitriou and Tim Roughgarden. 2008. Computing correlated equilibria in multi-player games. *Journal of the ACM (JACM)* 55, 3 (2008), 14.

[55] Robert W. Rosenthal. 1973. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory* 2, 1 (01 Dec 1973), 65–67. https://doi.org/10.1007/BF01737559

[56] Tim Roughgarden. 2015. Intrinsic Robustness of the Price of Anarchy. *J. ACM* 62, 5 (2015), 32:1–32:42. https://doi.org/10.1145/2806883

[57] Tim Roughgarden and Éva Tardos. 2000. How Bad is Selfish Routing?. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*. IEEE Computer Society, 93–102. https://doi.org/10.1109/SFCS.2000.892069

[58] Tim Roughgarden and Omri Weinstein. 2016. On the Communication Complexity of Approximate Fixed Points. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, Irit Dinur (Ed.). IEEE Computer Society, 229–238. https://doi.org/10.1109/FOCS.2016.32

[59] Aviad Rubinstein. 2016. Settling the complexity of computing approximate two-player Nash equilibria. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 258–265.

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

[61] Alejandro A. Schäffer and Mihalis Yannakakis. 1991. Simple Local Search Problems That are Hard to Solve. *SIAM J. Comput.* 20, 1 (1991), 56–87. https://doi.org/10.1137/0220004

[62] Alexander Skopalik and Berthold Vöcking. 2008. Inapproximability of pure Nash equilibria. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 355–364.

[63] Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. 2018. PPP-Completeness with Connections to Cryptography. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, Mikkel Thorup (Ed.). IEEE Computer Society, 148–158. https://doi.org/10.1109/FOCS.2018.00023

[64] Stephen A. Vavasis. 1993. Black-Box Complexity of Local Minimization. *SIAM J. Optim.* 3, 1 (1993), 60–80. https://doi.org/10.1137/0803004

[65] Adrian Vetta. 2002. Nash Equilibria in Competitive Societies, with Applications to Facility Location, Traffic Routing and Auctions. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*. IEEE Computer Society, 416. https://doi.org/10.1109/SFCS.2002.1181966

[66] E. B. Yanovskaya. 1968. Equilibrium points in polymatrix games. *Lithuanian Mathematical Journal* (1968).