The Complexity of Computing a Nash Equilibrium

Constantinos Daskalakis* University of California, Berkeley costis@cs.berkeley.edu

Paul W. Goldberg† University of Warwick pwg@dcs.warwick.ac.uk

Christos H. Papadimitriou‡ University of California, Berkeley christos@cs.berkeley.edu

ABSTRACT

We resolve the question of the complexity of Nash equilibrium by showing that the problem of computing a Nash equilibrium in a game with 4 or more players is complete for the complexity class PPAD. Our proof uses ideas from the recently-established equivalence between polynomial time solvability of normal form games and graphical games, establishing that these kinds of games can simulate a PPAD-complete class of Brouwer functions.

Categories and Subject Descriptors

F.2.0 [Analysis of Algorithms and Problem Complexity]:General

General Terms

Theory, Algorithms, Economics

Keywords

Complexity, Nash Equilibrium, PPAD-Completeness, Game Theory

1. INTRODUCTION

In 1951 Nash showed that every game has a Nash equilibrium [20]. The computational problem of finding such equilibria in polynomial time has remained open, and has come under increased scrutiny during the past two decades, see Section 2.2 for some references. The 2-player case seems a little easier, since linear programming-like techniques come into play and the solutions are guaranteed to be rational — Nash showed in his original paper that there are 3-player games with only irrational equilibria. The problem was known to belong to the class PPAD [23] of search problems with solutions guaranteed to exist by dint of a directed graph-theoretic argument but, unlike the problem of finding a Brouwer fixed point, it was not known to be complete for that class.

In this paper we show that the problem of computing Nash equilibria in games with 4 players is indeed PPAD-complete. Thus, a polynomial-time algorithm would imply a similar algorithm, e.g., for computing Brouwer fixed points, a problem for which quite strong lower bounds for large classes of algorithms are known [13]. Also, such an algorithm would have to fail to relativize with respect to the oracles in [1], for which PPAD has no polynomial-time algorithms.

Nash showed his result by reducing the existence of Nash equilibria to the existence of Brouwer fixed points. Given any game, he constructs a Brouwer function whose fixed points are precisely the equilibria of the game. In Nash's reduction, as well as in subsequent ones [10], the constructed Brouwer function is quite specialized, and this has led to speculation on whether the fixed points of such functions (and thus Nash equilibria) are easier to find than for general Brouwer functions. We answer this question in the negative by presenting a reduction in the opposite direction: Any (computationally presented) Brouwer function can be simulated by a game, so that Nash equilibria correspond to fixed points.

In a recent precursor of this paper [12], we describe certain reductions between equilibrium problems; in particular, finding a Nash equilibrium in an r-player game can be reduced to the same problem with 4 players, and to that in a graphical game (see Section 2.1 for the definition of a graphical game) with two strategies per player and maximum degree three. Thus, all these problems and their generalizations are PPAD-complete. Our present proof relies crucially on certain "arithmetical gadgets" invented in [12]: games that have the effect of adding, multiplying, and comparing real numbers.

The idea in our proof is this: Given a Brouwer function on the 3-dimensional cube (appropriately stylized and presented as a circuit), we simulate the coordinates by three players; each player has two strategies, so that the three mixed strategies are essentially a point in the cube. We construct a game in which the best responses of these three players implement the Brouwer function. This is done by decoding the coordinates into a bit vector, using the circuit to compute the value of the function, and then inducing the three players to add the appropriate increments to their mixed strategy. All this is done by a graphical game, ulti-

*Research supported by NSF ITR grant CCR-0121555 and a grant from Microsoft Research.

†Research supported by the EPSRC grant GR/T07343/01 "Algorithmics of Network-sharing Games". This work was begun while the author was visiting U. C. Berkeley.

‡Research supported by NSF ITR grant CCR-0121555 and a grant from Microsoft Research.

STOC'06, May21–23, 2006, Seattle, Washington, USA.

mately relying on the reduction in [12] to get to a 4-player normal form game.

There are many difficulties, of course. One has to do with *brittle comparators*. Our comparator gadget sets a number z to 0 if $x < y$, to 1 if $x > y$, *and to anything if* $x = y$ (it is not hard to see that no "robust" comparator gadget is possible). This means that decoding is flaky at the boundaries, and at these points the best response can be essentially arbitrary. We solve this by defining the best response to be an average over a "microlattice" around the point of interest, thus smoothing out any effects from boundaries of measure zero. Another problem is that the Brouwer problem, as defined in [23], would be very cumbersome to use in this reduction; thus, a side-product of our proof is a much cleaner canonical version of the computational problem associated with Brouwer's Fixed Point Theorem (actually, one that is, in a sense, half-way between that and Sperner's Lemma).

2. BACKGROUND

2.1 Games

A *game in normal form* has $r \geq 2$ players (indexed by p) and for each player p a finite set S_p of pure strategies. The set S of *pure strategy profiles* is the Cartesian product of the S_p's. We denote the set of all strategy profiles of players other than p by S_{-p}. Finally, for each $p \leq r$ and $s \in S$ we have a *payoff* or *utility* u_s^p — also occasionally denoted u_{js}^p for $j \in S_p$ and $s \in S_{-p}$.

A *mixed strategy* for player p is a distribution on S_p, that is, real numbers $x_j \geq 0$ for each strategy $j \in S_p$ such that $\sum_{j \in S_p} x_j = 1$. A set of r mixed strategies $\{x_j^p\}_{j \in S_p}, p = 1, \ldots, r$, is called a *(mixed) Nash equilibrium* if, for each p, $\sum_{s \in S} u_s^p x_s$ is maximized over all mixed strategies of p (where for a strategy profile $s = (s_1, \ldots, s_r) \in S$, we denote by x_s the product $x_{s_1}^1 \cdot x_{s_2}^2 \cdots x_{s_r}^r$). That is, a Nash equilibrium is a set of mixed strategies from which no player has a unilateral incentive to deviate. It is well-known (see, e.g., [22]) that the following is an equivalent condition for a set of mixed strategies to be a Nash equilibrium:

$$\sum_{s \in S_{-p}} u_{js}^p x_s > \sum_{s \in S_{-p}} u_{j's}^p x_s \Longrightarrow x_{j'}^p = 0.$$

More generally, a set of mixed strategies is an *ϵ-Nash equilibrium* for some $\epsilon > 0$ if the following holds:

$$\sum_{s \in S_{-p}} u_{js}^p x_s > \sum_{s \in S_{-p}} u_{j's}^p x_s + \epsilon \Longrightarrow x_{j'}^p = 0.$$

A game in normal form requires $r|S|$ numbers for its description — an exponential in the number of players amount of information. A *graphical game* [15] is defined in terms of an undirected graph $G = (V, E)$ together with a set of strategies S_v for each $v \in V$. We denote by $\mathcal{N}(v)$ the set consisting of v and v's neighbors in G, and by $S_{\mathcal{N}(v)}$ the set of all $|\mathcal{N}(v)|$-tuples of strategies, one from each vertex in $\mathcal{N}(v)$. In a graphical game, the utility of a vertex $v \in V$ only depends on the strategies of the vertices in $\mathcal{N}(v)$ so it can be represented by just $|S_{\mathcal{N}(v)}|$ numbers. In other words, a graphical game is a succinct representation of a multiplayer game, applicable when it so happens that the utility of each player only depends on a few other players.

2.2 Previous Work

Despite much interest, there have been very few complexity results (positive or negative) for the Nash equilibrium problem. Lipton and Markakis [17] study the algebraic properties of Nash equilibria, and point out that standard quantifier elimination algorithms can be used to solve them, but these are not polynomial-time in general. Papadimitriou and Roughgarden [24] show that, in the case of *symmetric* games, quantifier elimination results in polynomial algorithms for a broad range of parameters. Lipton, Markakis and Mehta [18] show that, if we only require an equilibrium that is best response within some accuracy ϵ, then a subexponential algorithm is possible. The two-player case, 2-NASH, can be solved by the simplex-like algorithm by Lemke and Howson [16]; this algorithm was recently shown to have exponential worst case running time [26]. If specific Nash equilibria are required, for example the Nash equilibrium with best social cost, the problem is typically NP-complete [11, 6].

In addition to [12], the papers [5, 27] have recently started to explore the computational complexity of games, via reductions between alternative types of games.

3. PPAD

3.1 Total Search Problems

We now review several definitions and results from the complexity theory of total functions, see [23] for a very similar, but not identical, formalism. A *search problem* $\mathcal{S}$ is a set of inputs $I_{\mathcal{S}} \subseteq \Sigma^*$ such that for each $x \in I_{\mathcal{S}}$ there is an associated set of solutions $\mathcal{S}_x \subseteq \Sigma^{|x|^k}$ for some integer k, such that for each $x \in I_{\mathcal{S}}$ and $y \in \Sigma^{|x|^k}$ whether $y \in \mathcal{S}_x$ is decidable in polynomial time (notice that this is precisely NP with an added emphasis on finding a witness).

For example, r-NASH is the search problem $\mathcal{S}$ in which each $x \in I_{\mathcal{S}}$ is an r-player game in normal form together with a binary integer A (the *accuracy specification*), and $\mathcal{S}_x$ is the set of $\frac{1}{A}$-Nash equilibria of the game. Similarly, d-GRAPHICAL NASH is the search problem with inputs the set of all graphical games with degree at most d, plus an accuracy specification, and solutions the corresponding approximate Nash equilibria.

A search problem is *total* if $\mathcal{S}_x \neq \emptyset$ for all $x \in I_{\mathcal{S}}$. For example, Nash's 1951 theorem [20] implies that r-NASH is total. Obviously, the same is true for d-GRAPHICAL NASH. The set of all total search problems is denoted TFNP.

Since TFNP is a "semantic" class (i.e., it has no generic complete problem), we explore its complexity via its important subclasses: PLS [14], PPP, PPA and PPAD [23]. In particular, PPAD is the class of all total search problems reducible to the following:

END OF THE LINE: Given two circuits S and P, each with n input bits and n output bits, such that $P(0^n) = 0^n \neq S(0^n)$, find an input $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$ or $S(P(x)) \neq x \neq 0^n$.

Intuitively, END OF THE LINE creates a directed graph with vertex set $\{0,1\}^n$ and an edge from x to y whenever both $y = S(x)$ and $x = P(y)$ (S and P stand for "successor candidate" and "predecessor candidate"). All nodes in this graph have indegree and outdegree at most one, and there is at least one source, namely 0^n, so there must be a sink. We seek either a sink, or a source other than 0^n. Thus, PPAD

(the initials stand for *polynomial parity argument, directed version*) is the class of all total functions whose totality is proven via the following simple combinatorial argument: "If a finite directed graph has an unbalanced node, then it must have another."

The following is shown in [23] by a reduction to END OF THE LINE via Brouwer's Theorem and Sperner's Lemma:

PROPOSITION 1. *r-NASH is in PPAD.*

A polynomially computable function f is a *polynomial-time reduction* from total search problem $\mathcal{S}$ to total search problem $\mathcal{T}$ if, for every input x of $\mathcal{S}$, $f(x)$ is an input of $\mathcal{T}$, and furthermore there is another polynomial time computable function g such that for every $y \in \mathcal{T}_{f(x)}$, $g(y) \in \mathcal{S}_x$. In [12] the following is shown:

THEOREM 1. *There are polynomial-time reductions from r-NASH and d-GRAPHICAL NASH, for any $r, d \geq 2$, to both 4-NASH and 3-GRAPHICAL NASH.*

A search problem $\mathcal{S}$ in PPAD is called *PPAD-complete* if all problems in PPAD reduce to it. Obviously, END OF THE LINE is PPAD-complete; our main result in this paper (Theorem 3) states that so are 4-NASH and 3-GRAPHICAL NASH.

3.2 An Interesting PPAD-Complete Problem

In the proof of our main result we use a problem we call 3-DIMENSIONAL BROUWER, which is a very simplified version of a problem shown PPAD-complete in [23]. We are given a stylized Brouwer function ϕ on the 3-dimensional unit cube, defined in terms of its values at the centers of 2^{3n} cubelets with side 2^{-n}. At the center c_{ijk} of the cubelet K_{ijk} defined as

$$K_{ijk} = \{(x,y,z) : \; i \cdot 2^{-n} \leq x \leq (i+1) \cdot 2^{-n}, \\ j \cdot 2^{-n} \leq y \leq (j+1) \cdot 2^{-n}, \\ k \cdot 2^{-n} \leq z \leq (k+1) \cdot 2^{-n}\},$$

where i, j, k are integers in $[2^n]$, the value of ϕ is $\phi(c_{ijk}) = c_{ijk} + \delta_{ijk}$, where δ_{ijk} is one of the following four vectors (also referred to as colors):

- $\delta_1 = (\alpha, 0, 0)$
- $\delta_2 = (0, \alpha, 0)$
- $\delta_3 = (0, 0, \alpha)$
- $\delta_0 = (-\alpha, -\alpha, -\alpha)$

Here $\alpha > 0$ is much smaller than the cubelet side, say 2^{-2n}.

In the actual continuous Brouwer function (which does not concern us in this problem as defined) the value of ϕ near the boundaries of the cubelets would be determined by interpolation — there are many simple ways to do this, and the precise method is of no importance to our discussion.

Thus, to compute ϕ at the centers of the cubelet K_{ijk} we only need to know which of the four displacements to add. This is computed by a circuit C (which is the only input to the problem) with $3n$ input bits and 2 output bits; $C(i,j,k)$ is the index r such that, if c is the center of cubelet K_{ijk}, $\phi(c) = c + \delta_r$. C is such that $C(0,j,k) = 1$, $C(i,0,k) = 2$, $C(i,j,0) = 3$ (with conflicts resolved arbitrarily) and $C(2^n - 1, j, k) = C(i, 2^n - 1, k) = C(i, j, 2^n - 1) = 0$, so that the function ϕ maps the boundary to the interior of the cube. A vertex of a cubelet is called *panchromatic* if among the eight cubelets adjacent to it there are four that have all four increments $\delta_0, \delta_1, \delta_2, \delta_3$.

3-DIMENSIONAL BROUWER is thus the following problem: Given a circuit C as described above, find a panchromatic vertex. The relationship with Brouwer fixpoints (as promised by Brouwer's theorem) is that any natural interpolation rule ensures that fixpoints only ever occur in the vicinity of a panchromatic vertex; elsewhere the displacements cannot cancel each other out. In fact, we can show the following.

THEOREM 2. *3-DIMENSIONAL BROUWER is PPAD-complete.*

PROOF. The reduction is from END OF THE LINE. Given circuits S and P with n inputs and outputs, as prescribed in this problem, we shall construct an equivalent instance of 3-DIMENSIONAL BROUWER, that is, another circuit C with $3m = 3(n+3)$ inputs and two outputs that computes the color of each cubelet of side 2^{-m}, that is to say, the index i such that δ_i is the correct increment of the Brouwer function at the center of the cubelet encoded into the $3m$ bits of the input. We shall first describe the Brouwer function ϕ explicitly, and then argue that it can be computed by a circuit.

Our description of ϕ proceeds as follows: We shall first describe a 1-dimensional subset L of the 3-dimensional unit cube, intuitively an embedding of the path-like directed graph $G_{S,P}$ implicitly given by S and P. Then we shall describe the 4-coloring of the 2^{3m} cubelets based on the description of L. Finally, we shall argue that colors are easy to compute locally, and that fixed points correspond to endpoints other than 0^n of the line.

We assume that the graph $G_{S,P}$ is such that for each edge (u,v), one of the vertices is even (ends in 0) and the other is odd; this is easy to guarantee (we could add a dummy input and output bit to S and P that is always flipped.)

L will be orthonormal, that is, each of its segments will be parallel to one of the axes. Let $u \in \{0,1\}^n$ be a vertex of $G_{S,P}$. By $[u]$ we denote the integer between 0 and $2^n - 1$ whose binary representation is u; all coordinates of endpoints of segments are integer multiples of 2^{-m}, a factor that we omit. Associated with u there are two line segments of length 4 of L. The first, called the *principal* segment of u, has endpoints $u_1 = (8[u] + 2, 3, 3)$ and $u_1' = (8[u] + 6, 3, 3)$. The other *auxiliary* segment has endpoints $u_2 = (3, 8[u]+2, 2^m - 3)$ and $u_2' = (3, 8[u]+6, 2^m - 3)$. Informally, these segments form two dashed lines (each segment being a dash) that run along two edges of the cube and slightly in its interior (see Figure 1).

Now, for every vertex u of $G_{S,P}$, we connect u_1' to u_2 by a line with three straight segments, with joints $u_3 = (8[u] + 6, 8[u] + 2, 3)$ and $u_4 = (8[u] + 6, 8[u] + 2, 2^m - 3)$. Finally, if there is an edge (u,v) in $G_{S,P}$, we connect u_2' to v_1 by a broken line with breakpoints $u_5 = (8[v] + 2, 8[u] + 6, 2^m - 3)$ and $u_6 = (8[v]+2, 8[u]+6, 3)$. This completes the description of the line L if we do the following perturbation: exceptionally, the principal segment of $u = 0$ has endpoints $0_1 = (2,2,2)$ and $0_1' = (6,2,2)$ and the corresponding joint is $0_3 = (6,2,2)$.

It is easy to see that L traverses the interior of the cube without ever "nearly crossing itself"; that is, two points p, p' of L are closer than $4 \cdot 2^{-m}$ in Euclidean distance only if they

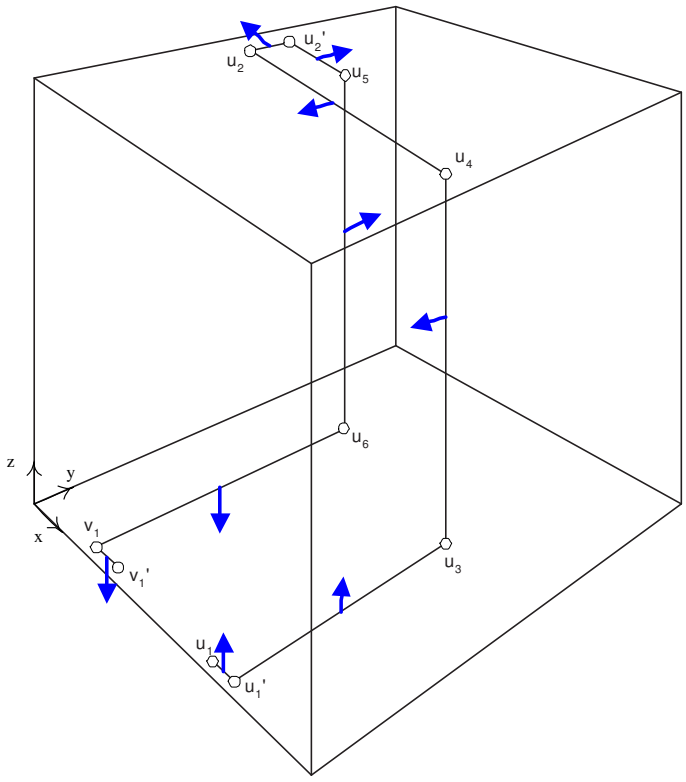

Figure 1: The orthonormal path connecting nodes (u,v); the arrows indicate the orientation of colors surrounding the path.

are connected by a part of L that has length $8 \cdot 2^{-m}$ or less. (This is important in order for the coloring described below of the cubelets surrounding L to be well-defined.) To check this, just notice that segments of different types (e.g., $[u_3, u_4]$ and $[v_2', v_5]$) come close only if they share an endpoint; segments on the $z = 3$ plane are parallel and at least 4 apart; and segments parallel to the z axis differ by at least 4 in either their x or y coordinates.

We now describe the coloring of the 2^{3m} cubelets by four colors corresponding to the four increments. As required for a 3-DIMENSIONAL BROUWER circuit, the color of any cubelet K_{ijk} where any one of i, j, k is $2^m - 1$, is 0. Given that, any other cubelet with $i = 0$ gets color 1; with this fixed, any other cubelet with $j = 0$ gets color 2, while the remaining cubelets with $k = 0$ get color 3. Having colored the boundaries, we now have to color the interior cubelets. An interior cubelet is always colored 0 *unless one of its vertices is a point of the interior of line* L, in which case it is colored by one of the three other colors in a manner to be explained shortly. Intuitively, at each point of the line L, starting from $(2, 2, 2)$ (the beginning of the principle segment of the string $u = 0^n$) the line L is "protected" from color 0 from all 4 sides. As a result, the only place where the four colors can meet is vertex u_2' or $u_1, u \neq 0^n$, where u is an end of the line...

In particular, near the beginning of L at $(2, 2, 2)$ the 27 cubelets K_{ijk} with $i, j, k \leq 2$ are colored as shown in Figure 2. From then on, for any length-1 segment of L of the form $[(x, y, z), (x', y', z')]$ consider the four cubelets containing this edge. Two of these cubelets are colored 3, and the other two are colored 1 and 2, in this order clockwise (from the point of view of an observer at (x, y, z)) as L proceeds from $(2, 2, 2)$ on. The remaining cubelets touching L are the ones at the joints where L turns. Each of these cubelets, a total of two per turn, takes the color of the two other cubelets adjacent to L with which it shares a face.

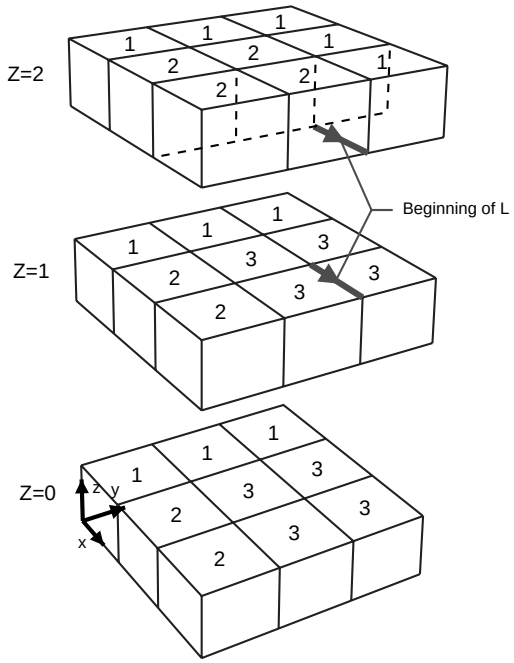

Figure 2: The 27 cubelets around the beginning of line L.

Now it only remains to describe, for each line segment $[a, b]$ of L, the direction d in which the two cubelets that are colored 3 lie. The rules are these (in Figure 1 the directions d are shown as arrows):

- If $[a, b] = [u_1, u_1']$ then $d = (0, 0, -1)$ if u is even and $d = (0, 0, 1)$ if u is odd.
- If $[a, b] = [u_1', u_3]$ then $d = (0, 0, -1)$ if u is even and $d = (0, 0, 1)$ if u is odd.
- If $[a, b] = [u_3, u_4]$ then $d = (0, 1, 0)$ if u is even and $d = (0, -1, 0)$ if u is odd.
- If $[a, b] = [u_4, u_2]$ then $d = (0, 1, 0)$ if u is even and $d = (0, -1, 0)$ if u is odd.
- If $[a, b] = [u_2, u_2']$ then $d = (1, 0, 0)$ if u is even and $d = (-1, 0, 0)$ if u is odd.
- If $[a, b] = [u_2', u_5]$ then $d = (0, -1, 0)$ if u is even and $d = (0, 1, 0)$ if u is odd.
- If $[a, b] = [u_5, u_6]$ then $d = (0, -1, 0)$ if u is even and $d = (0, 1, 0)$ if u is odd.

- If $[a, b] = [u_6, v_1]$ then $d = (0, 0, 1)$ if u is even and $d = (0, 0, -1)$ if u is odd.

This completes the description of the construction. Notice that, for this to work, we need our assumption that edges in $G_{S,P}$ go between odd and even nodes. Regarding the alternating orientation of colored cubelets around L, note that we could not simply introduce "twists" to make them always point in (say) direction $d = (0, 0, -1)$ for all (u_1, u'_1). That would create a panchromatic vertex at the location of a twist.

The result now follows from the following two claims:

1. A point in the cube is panchromatic in the described coloring if and only if it is
   (a) an endpoint u'_2 of a sink vertex, or
   (b) u_1 of a source vertex $u \neq 0^n$ of $G_{S,P}$
2. A circuit C can be constructed in time polynomial in $|S| + |P|$, which computes, for each triple of binary integers $i, j, k < 2^m$, the color of cubelet K_{ijk}.

Regarding the first claim, the endpoint u'_2 of a sink vertex, or the endpoint u_1 of a source vertex other than 0^n, will be a point where L meets color 0, hence a panchromatic vertex. There is no alternative way that L can meet color 0.

Regarding the second claim, circuit C is doing the following. $C(0, j, k) = 1$, $C(i, 0, k) = 2$ for $i > 0$, $C(i, j, 0) = 3$ for $i, j > 0$. Then by default, $C(i, j, k) = 0$. However the following tests yield alternative values for $C(i, j, k)$, for cubelets adjacent to L. $LSB(x)$ denotes the least significant bit of x, equal to 1 if x is odd, 0 if x is even and undefined if x is not an integer.

For example, a $(u'_1, u_3), u \neq 0$ segment is given by:

1. If $k = 2$ and $i = 8x + 5$ and $LSB(x) = 1$ and $j \in \{3, \ldots, 8x + 2\}$ then $C(i, j, k) = 2$.
2. If $k = 2$ and $i = 8x + 6$ and $LSB(x) = 1$ and $j \in \{2, \ldots, 8x + 2\}$ then $C(i, j, k) = 1$.
3. If $k = 3$ and ($i = 8x+5$ or $i = 8x+6$) and $LSB(x) = 1$ and $j \in \{2, \ldots, 8x + 1\}$ then $C(i, j, k) = 3$.
4. If $k = 2$ and ($i = 8x+5$ or $i = 8x+6$) and $LSB(x) = 0$ and $j \in \{2, \ldots, 8x + 2\}$ then $C(i, j, k) = 3$.
5. If $k = 3$ and $i = 8x + 5$ and $LSB(x) = 0$ and $j \in \{3, \ldots, 8x + 1\}$ then $C(i, j, k) = 1$.
6. If $k = 3$ and $i = 8x + 6$ and $LSB(x) = 0$ and $j \in \{2, \ldots, 8x + 1\}$ then $C(i, j, k) = 2$.

A (u'_2, u_5) segment uses the circuits P and S, and in the case $LSB(x) = 1$ (where x is derived from j) is given by:

1. If ($k = 2^m - 3$ or $k = 2^m - 4$) and $j = 8x + 6$ and $S(x) = x'$ and $P(x') = x$ and $i \in \{2, \ldots, 8x' + 2\}$ then $C(i, j, k) = 3$.
2. If $k = 2^m - 3$ and and $j = 8x + 5$ and $S(x) = x'$ and $P(x') = x$ and $i \in \{3, \ldots, 8x' + 2\}$ then $C(i, j, k) = 1$.
3. If $k = 2^m - 4$ and $j = 8x + 5$ and $S(x) = x'$ and $P(x') = x$ and $i \in \{2, \ldots, 8x' + 1\}$ then $C(i, j, k) = 2$.

The other segments are done in a similar way, and so the second claim follows. □

4. THE MAIN REDUCTION

We now prove our main result:

THEOREM 3. 4-NASH *is PPAD-complete.*

PROOF. The reduction is from 3-DIMENSIONAL BROUWER. Given a circuit C with $3n$ input bits describing a Brouwer function as in the definition of 3-DIMENSIONAL BROUWER, we shall construct a graphical game $\mathcal{G}$, with maximum degree three, that simulates it. Since we know (Theorem 1) that graphical games reduce to 4-NASH, this completes the proof.

The graphical game $\mathcal{G}$ is *binary* in that each vertex v in it has two strategies, and thus, at equilibrium, it represents a real number in $[0, 1]$ denoted $\mathbf{p}[v]$. (Letting 0 and 1 denote the strategies, $\mathbf{p}[v]$ is the probability that v plays 1.) There are three distinguished nodes v_x, v_y, and v_z representing the coordinates of the Brouwer problem. We next define certain useful gadgets which allow us to make arithmetic operations, variants of the ones described in [12].

Notation: By $x = y \pm \epsilon$ we mean $y - \epsilon \leq x \leq y + \epsilon$.

LEMMA 1. *There are binary graphical games $\mathcal{G}_\zeta$, where ζ is any rational in $[0, 1]$, $\mathcal{G}_=, \mathcal{G}_+, \mathcal{G}_-, \mathcal{G}_*, \mathcal{G}_<$ with four players a, b, c, d such that in all games, the payoffs of a and b do not depend on the choices of the other vertices c, d, and*

1. *at any ϵ-Nash equilibrium in $\mathcal{G}_\zeta$ we have $\mathbf{p}[d] = \zeta \pm \epsilon$;*
2. *at any ϵ-Nash equilibrium in $\mathcal{G}_=$ we have $\mathbf{p}[d] = \mathbf{p}[a] \pm \epsilon$;*
3. *at any ϵ-Nash equilibrium in $\mathcal{G}_+$ we have $\mathbf{p}[d] = \min\{1, \mathbf{p}[a] + \mathbf{p}[b]\} \pm \epsilon$;*
4. *at any ϵ-Nash equilibrium in $\mathcal{G}_-$ in which $\mathbf{p}[a] \geq \mathbf{p}[b]$, we have $\mathbf{p}[d] = \mathbf{p}[a] - \mathbf{p}[b] \pm \epsilon$;*
5. *at any ϵ-Nash equilibrium in $\mathcal{G}_*$ we have $\mathbf{p}[d] = \mathbf{p}[a] \cdot \mathbf{p}[b] \pm \epsilon$;*
6. *at any ϵ-Nash equilibrium in $\mathcal{G}_<$ we have $\mathbf{p}[d] = 1$ if $\mathbf{p}[a] < \mathbf{p}[b] - \epsilon$ and $\mathbf{p}[d] = 0$ if $\mathbf{p}[a] > \mathbf{p}[b] + \epsilon$.*

PROOF. This is a simple extension of Propositions 1-3 of [12]. Player c is needed for $\mathcal{G}_=$, $\mathcal{G}_+$, $\mathcal{G}_-$ and $\mathcal{G}_*$; its role is to mediate between a, b and d.

The constructions are straightforward, for example for Item 2, the payoffs in $\mathcal{G}_=$ are

Payoffs to d :

	c plays 0	c plays 1
d plays 0	0	1
d plays 1	1	0

Payoffs to c:

		d plays 0	d plays 1
c plays 0	a plays 0	0	0
	a plays 1	1	1

		d plays 0	d plays 1
c plays 1	a plays 0	0	1
	a plays 1	0	1

If c plays 1, then the expected payoff to c is $\mathbf{p}[d]$, and if c plays 0 then the expected payoff to c is $\mathbf{p}[a]$. Therefore, in a ϵ-Nash equilibrium of $\mathcal{G}_=$, if $\mathbf{p}[d] > \mathbf{p}[a] + \epsilon$ then $\mathbf{p}[c] = 1$.

However, note also that if $\mathbf{p}[c] = 1$ then $\mathbf{p}[d] = 0$. (Payoffs to d make it prefer to disagree with c.) Consequently, $\mathbf{p}[d]$ cannot be strictly larger than $\mathbf{p}[a] + \epsilon$.

Similarly, if $\mathbf{p}[d] < \mathbf{p}[a] - \epsilon$ then $\mathbf{p}[c] = 0$, which implies that $\mathbf{p}[d] = 1$ (again since d has the biggest payoff by disagreeing with c). Hence $\mathbf{p}[d]$ cannot be less than $\mathbf{p}[a] - \epsilon$.

For Item 3, the payoffs in $\mathcal{G}_+$ are

Payoffs to d :

	c plays 0	c plays 1
d plays 0	0	1
d plays 1	1	0

Payoffs to c:

		b plays 0	b plays 1
c plays 0	a plays 0	0	1
	a plays 1	1	2

c plays 1	d plays 0	0
	d plays 1	1

If c plays 1, then the expected payoff to c is $\mathbf{p}[d]$, and if c plays 0 then the expected payoff to c is $\mathbf{p}[a]+\mathbf{p}[b]$. Therefore, in a ϵ-Nash equilibrium of $\mathcal{G}_+$, if $\mathbf{p}[d] > \mathbf{p}[a] + \mathbf{p}[b] + \epsilon$ then $\mathbf{p}[c] = 1$.

However, note from the payoffs to d that if $\mathbf{p}[c] = 1$ then $\mathbf{p}[d] = 0$. Consequently, $\mathbf{p}[d]$ cannot be strictly larger than $\mathbf{p}[a] + \mathbf{p}[b] + \epsilon$.

Similarly, if $\mathbf{p}[d] < \mathbf{p}[a] + \mathbf{p}[b] - \epsilon$ then due to the payoffs to c we have $\mathbf{p}[c] = 0$. This in turn implies that $\mathbf{p}[d] = 1$ (since d has the biggest payoff by disagreeing with c). Hence $\mathbf{p}[d]$ cannot be less than $\min(1, \mathbf{p}[a] + \mathbf{p}[b] - \epsilon)$.

$\mathcal{G}_<$ is as follows: player d receives a payoff of 1 if d plays 0 and a plays 1, and d receives a payoff of 1 if d plays 1 and b plays 1, otherwise d received a payoff of 0.

Equivalently, d receives an expected payoff of $\mathbf{p}[a]$ if d plays 0, and receives $\mathbf{p}[b]$ of d plays 1. It can be verified that this implements $\mathcal{G}_<$. □

Notice that, in $\mathcal{G}_<$, $\mathbf{p}[d]$ is arbitrary if $\mathbf{p}[a]$ is close to $\mathbf{p}[b]$, hence we call it the *brittle* comparator. As an aside, it is not hard to see that a robust comparator, one in which d is guaranteed at an exact Nash equilibrium to be, say, 0 if $\mathbf{p}[a] = \mathbf{p}[b]$, cannot exist, since it could be used to produce a simple graphical game with no Nash equilibrium, contradicting Nash's theorem.

To continue the reduction, the graphical game $\mathcal{G}$ will contain the following vertices.

- the three coordinate vertices v_x, v_y, v_z,
- for $i \in \{1, 2, \ldots, n\}$, vertices $v_{b_i(x)}$, $v_{b_i(y)}$ and $v_{b_i(z)}$, whose $\mathbf{p}$-values represent the i-th most significant bit of $\mathbf{p}[v_x]$, $\mathbf{p}[v_y]$, $\mathbf{p}[v_z]$,
- for $i \in \{1, 2, \ldots, n\}$, vertices v_{x_i}, v_{y_i} and v_{z_i}, whose $\mathbf{p}$-values represent the fractional number resulting from subtracting from $\mathbf{p}[v_x]$, $\mathbf{p}[v_y]$, $\mathbf{p}[v_z]$ the fractional numbers corresponding to the $i-1$ most significant bits of $\mathbf{p}[v_x]$, $\mathbf{p}[v_y]$, $\mathbf{p}[v_z]$ respectively.

We can extract these values by computing the binary representation of $\lfloor \mathbf{p}[v_x] 2^n \rfloor$ and similarly for v_y and v_z — that is, the binary representations of the integers i, j, k such that $(x, y, z) = (\mathbf{p}[v_x], \mathbf{p}[v_y], \mathbf{p}[v_z])$ lies in the cubelet K_{ijk}. This is done by a graphical game that simulates, using the arithmetical gadgets of Lemma 1, the following algorithm (by $< (a, b)$ we mean the brittle comparator of a and b):

$x_1 = x$ (short for $\mathbf{p}[v_{x_1}] = \mathbf{p}[v_x]$);
for $i = 1, \ldots, n$ *do:*
$\{b_i(x) :=< (2^{-i}, x_i);\ x_{i+1} := x_i - b_i(x) \cdot 2^{-i}\}$
Similarly for y and z.

This is accomplished in $\mathcal{G}$ by connecting these nodes as prescribed by Lemma 1, so that $\mathbf{p}[v_{x_i}], \mathbf{p}[v_{b_i(x)}]$, etc. approximate the value of x_i, $b_i(x)$ etc. as computed by the above algorithm. The following lemma (when applied with $m = n$) shows that this device properly decodes the first n bits of the binary expansion of $x = \mathbf{p}[v_x]$, as long as x is not too close to a multiple of 2^{-n} (suppose $\epsilon << 2^{-n}$ to be fixed later).

LEMMA 2. *For $m \le n$, if $\sum_{i=1}^{m} b_i 2^{-i} + 2m\epsilon < x < \sum_{i=1}^{m} b_i 2^{-i} + 2^{-m} - 2m\epsilon$ for some $b_1, \ldots, b_m \in \{0, 1\}$, then at any ϵ-Nash equilibrium of $\mathcal{G}$, $\mathbf{p}[v_{b_m(x)}] = b_m$, and $\mathbf{p}[v_{x_{m+1}}] = x - \sum_{i=1}^{m} b_i 2^{-i} \pm 2m\epsilon$.*

PROOF. Induction on m. It is trivial for $m = 1$, so suppose it holds up to m. Then we know that $\mathbf{p}[v_{x_{m+1}}]$ is not within ϵ of a multiple of 2^{-m}, and therefore $\mathbf{p}[v_{b_{m+1}(x)}]$ will hold the correct value of the next bit. Finally, the multiplication and subtraction needed to compute x_{m+2} will each introduce an additional error of ϵ, concluding the induction. □

The above is repeated for y and z to obtain i, j, k, that is, $3n$ vertices of the graph of $\mathcal{G}$ whose $\mathbf{p}$ values correspond to the bits of i, j, k (assuming that x, y, z are all at least $2n\epsilon$ away from any multiple of 2^{-n}). Once we have the binary representations of i, j, k, we can feed them into another part of $\mathcal{G}$ that simulates the circuit C. We could simulate the circuit by having nodes that represent gates, using addition (with ceiling 1) to simulate *or*, multiplication for *and*, and $1-x$ for negation. However, there is a simpler way, one that avoids the complications related to accuracy, to simulate Boolean functions under the assumption that the inputs are 0 or 1:

LEMMA 3. *There are binary graphical games $\mathcal{G}_\vee, \mathcal{G}_\wedge, \mathcal{G}_\neg$ with two input players a, b (one input player a for $\mathcal{G}_\neg$) and an output player c such that the payoffs of a and b do not depend on the choices of c, and, at any ϵ-Nash equilibrium with $\epsilon < 1/4$ in which $\mathbf{p}[a], \mathbf{p}[b] \in \{0, 1\}$, $\mathbf{p}[c]$ is also in $\{0, 1\}$, and is in fact the result of applying the corresponding Boolean function to the inputs.*

PROOF. These games are in the same spirit as $G_<$. In $G_\vee$, for example, the payoff to c is $1/2$ if it plays 0; if c plays 1 its payoff is 1 if at least one of a, b plays 1, and it is 0 if they both play 0. Similarly for $G_\wedge$ and $G_\neg$ □

Thus in addition to the part of $\mathcal{G}$ that computes the bits $\mathbf{p}[v_{b_i(x)}]$ etc. of i, j, k we have a part that simulates C, containing one vertex for each gate of C, and such that, in any ϵ-approximate equilibrium in which all the $\mathbf{p}[v_{b_i(x)}]$'s are $0 - 1$, the vertices corresponding to the outputs of C also play pure strategies, and furthermore these strategies correspond correctly to the outputs of C.

In fact, it is convenient to assume that the output of C is a little more explicit: C computes six bits Δx^+,Δx^-,

Δy^+,Δy^-, Δz^+,Δz^- such that at most one of $\Delta x^+, \Delta x^-$ is 1, and at most one of $\Delta y^+, \Delta y^-$ is 1, and similarly for z, and the increment of the Brouwer function at the center of K_{ijk} is $\alpha \cdot (\Delta x^+ - \Delta x^-, \Delta y^+ - \Delta y^-, \Delta z^+ - \Delta z^-)$, one of $\delta_0, \delta_1, \delta_2, \delta_3$ as defined above. It would seem that all we have to do now is close the loop by incentivizing (using the G_+ device of Lemma 1) the v_x, v_y and v_z vertices to increment their $\mathbf{p}$ values by the appropriate amounts.

But, as we mentioned above, there is a problem: Because of the brittle comparators, at the boundaries of the cubelets the vertices that should represent the values of the bits of i, j, k hold in fact arbitrary reals, and therefore so do the output vertices of C, the Δx^+ etc, and this noise in the calculation can create spurious Nash equilibria. Suppose for example that (x, y, z) lies on the boundary between two cubelets that have color 1. Then there ought not to be a Nash equilibrium with $\mathbf{p}[v_x] = x$, $\mathbf{p}[v_y] = y$, $\mathbf{p}[v_z] = z$. We want that if $\mathbf{p}[v_x] = x$, $\mathbf{p}[v_y] = y$, $\mathbf{p}[v_z] = z$, then $\mathbf{p}[v_x], \mathbf{p}[v_y], \mathbf{p}[v_z]$ should have the incentive to move in direction δ_1, so that v_x prefers to increase $\mathbf{p}[v_x]$. However, on a boundary between two cubelets, some of the "bit values" that get loaded into $v_{b_i(x)}$, could be other than 0 and 1, and then there is nothing we can say about the output of the circuit that processes these values.

To overcome this difficulty, we resort to the following *averaging maneuver:* We repeat the above computation not just for the point (x, y, z), but for all $M = (2m+1)^3$ points of the form $(x + p \cdot \alpha, y + q \cdot \alpha, z + s \cdot \alpha)$ for $-m \leq p, q, s \leq m$, where m is a large enough constant to be fixed later (we show below that $m = 20$ is sufficient). This is done by M copies of the device described above, yielding $6M$ output bits $\Delta x_1^+, \ldots, \Delta z_M^-$. A final part of $\mathcal{G}$ calculates the vector

$$(\delta x, \delta y, \delta z) = \frac{\alpha}{M} \sum_{t=1}^{M} (\Delta x_t^+ - \Delta x_t^-, \Delta y_t^+ - \Delta y_t^-, \Delta z_t^+ - \Delta z_t^-) \tag{1}$$

the average increment of all M points (this is done using the arithmetic gadgets of Lemma 1).

We can now close the loop by inserting addition gadgets that force, at equilibrium, $\mathbf{p}[v_x]$ to be $x + \delta x$, where by x we mean the value of a vertex that is a copy of v_x, and similarly for v_y and v_z. This concludes the reduction; it is clear that it can be carried out in polynomial time.

Our proof is concluded by the following claim. For the following lemma we choose $\epsilon = \alpha^2$. Recall from our definition of 3-DIMENSIONAL BROUWER that we chose $\alpha = 2^{-2n}$.

LEMMA 4. *In any ϵ-Nash equilibrium of the game $\mathcal{G}$, one of the vertices of the cubelets that contain $(\mathbf{p}[v_x], \mathbf{p}[v_y], \mathbf{p}[v_z])$ is panchromatic.*

PROOF. We start by pointing out a simple property of the increments $\delta_0, \ldots, \delta_3$:

LEMMA 5. *Suppose that for nonnegative integers $k_0, \ldots, k_3$ all three coordinates of $\sum_{i=0}^{3} k_i \delta_i$ are smaller in absolute value than $\frac{\alpha K}{5}$ where $K = \sum_{i=0}^{3} k_i$. Then all four k_i are positive.*

PROOF. For the sake of contradiction, suppose that $k_1 = 0$. It follows that $k_0 < K/5$ (otherwise the negative x coordinate would be too large), and thus one of k_2, k_3 is larger than $2K/5$, which makes the corresponding coordinate too large, a contradiction. Similarly if $k_2 = 0$ or $k_3 = 0$. Finally, if $k_0 = 0$ then one of k_1, k_2, k_3, and the associated component, is at least $K/3$, again a contradiction. □

The small value of ϵ relative to α first implies that, for direction x, at most one of the values $x + p \cdot \alpha$ can be $2n\epsilon$-close to a multiple of 2^{-n}, and similarly for y and z. Hence, from among the $M = (2m+1)^3$ evaluations, all but at most $3(2m+1)^2$, or at least $K = (2m-2)(2m+1)^2$, compute legitimate Δx^+ etc. values. The corresponding K terms of summation (1) add up to $\frac{1}{M} \sum_{i=0}^{3} k_i \delta_i$ for some nonnegative integers $k_0, \ldots, k_3$ adding up to K (recall that from Lemma 3 these K evaluations of the circuit will produce binary outputs). The remaining terms can add up to a vector with each coordinate bounded from above in absolute value by $\frac{\alpha}{M} 6(2m+1)^2$. Thus, for $\mathcal{G}$ to be in a ϵ-Nash equilibrium, $\sum_{i=0}^{3} k_i \delta_i$ must add up to a vector with each coordinate bounded in absolute value by $\alpha 6(2m+1)^2 + 4M^2\epsilon$ (taking into account the error introduced when computing the average of equation (1)). By choosing $m = 20$, the bound becomes less than $\alpha K/5$, and so Lemma 5 applies. It follows that among the results of the K computations of the increments, all four $\delta_0, \ldots, \delta_3$ appeared. This implies that among the corners of the cubelets containing (x, y, z) there must be one panchromatic corner, completing the proof of Lemma 4. □

To conclude the proof of Theorem 3, if we find any ϵ-Nash equilibrium of $\mathcal{G}$, Lemma 4 has shown that by reading off the first n binary digits of $\mathbf{p}[v_x]$, $\mathbf{p}[v_y]$ and $\mathbf{p}[v_z]$ we obtain a solution to the corresponding instance of 3-DIMENSIONAL BROUWER. □

5. OPEN PROBLEMS

The work presented in this paper leaves open the complexity of computing a Nash equilibrium in 2 and 3 player games. Since the paper first appeared considerable work has been done in these directions. Specifically, Chen and Deng [2] and, independently, Daskalakis and Papadimitriou [8], prove that the problem of computing a Nash equilibrium in 3-player games is also PPAD-complete. The proofs are very different: Daskalakis and Papadimitriou [8] use a new graphical game for addition and multiplication whose graph is 3-colorable in the sense that the reduction of [12] specifies and, therefore, can be embedded in a 3-player normal form game. The proof of Chen and Deng [2] exploits a discontinuity in the payoff matrices of the addition game of Lemma 1, which allows its embedding into a 3-player normal form game using the technique of [12]. In fact, this discontinuity is exploited even further in [3], yielding the PPAD-completeness of computing a Nash equilibrium in 2-player games as well. Specifically, the proof of [3] is essentially the same as the one presented in the proof of Theorem 3 but with the following twist: instead of using the generic reduction of [12] to embed the resulting graphical game into a normal form game, a direct embedding is defined that, although mimics the embedding of [12], it exploits the structure of the payoff matrices of the arithmetical graphical games defined in Lemma 1 to reduce the number of players of the resulting normal form game.

A most important problem left open by our work is that of computing approximate Nash equilibria with less than exponential accuracy. Chen, Deng and Teng recently proved

that there exists an inverse polynomial ϵ for which computing a Nash equilibrium in a 2-player game remains PPAD-complete [4]. What happens for larger ϵ's is an intriguing and open question.

Besides normal-form games, our work settles the complexity of Nash equilibria in graphical games. Another important problem is the complexity of the same problem in other classes of succinctly representable games with many players; for example, are these problems even in PPAD? (It is typically easy to see that they cannot be easier than the normal-form problem.) In [7] we give a general sufficient condition, satisfied by all known succinct representations of games, for membership of the Nash equilibrium problem in the class PPAD.

6. REFERENCES

[1] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo, T. Pitassi. "The Relative Complexity of NP Search Problems," *J. Comput. Syst. Sci. 57,* 1, pp. 13–19, 1998.

[2] X. Chen and X. Deng. "3-NASH is PPAD-Complete," *Electronic Colloquium in Computational Complexity TR05-134*, 2005.

[3] X. Chen and X. Deng. "Settling the Complexity of 2-Player Nash-Equilibrium," *Electronic Colloquium in Computational Complexity TR05-140*, 2005.

[4] X. Chen, X. Deng and S. Teng. "Computing Nash Equilibria: Approximation and Smoothed Complexity," *arXiv report*, 2006.

[5] B. Codenotti, A. Saberi, K. Varadarajan and Y. Ye. "Leontief Economies Encode Nonzero Sum Two-Player Games," *Proceedings of SODA*, 2006.

[6] V. Conitzer and T. Sandholm. "Complexity Results about Nash Equilibria," *Proceedings of IJCAI*, 2003.

[7] C. Daskalakis, A. Fabrikant and C. H. Papadimitriou. "The Game World is Flat: The Complexity of Nash Equilibria in Succinct Games," *To appear*, 2006.

[8] C. Daskalakis and C. H. Papadimitriou. "Three-Player Games Are Hard," *Electronic Colloquium in Computational Complexity TR05-139*, 2005.

[9] A. Fabrikant, C.H. Papadimitriou and K. Talwar. "The Complexity of Pure Nash Equilibria," *Proceedings of STOC*, 2004.

[10] J. Geanakoplos. "Nash and Walras Equilibrium via Brouwer," *Economic Theory, 21*, 2003.

[11] I. Gilboa and E. Zemel. "Nash and correlated equilibria: Some complexity considerations," *Games and Economic Behavior*, 1989.

[12] P. W. Goldberg and C. H. Papadimitriou. "Reducibility Among Equilibrium Problems," *Proceedings of STOC*, 2006.

[13] M. Hirsch, C. H. Papadimitriou and S. Vavasis. "Exponential Lower Bounds for Finding Brouwer Fixpoints," *J. Complexity 5*, pp. 379–416, 1989.

[14] D. S. Johnson, C. H. Papadimitriou and M. Yannakakis, "How Easy is Local Search?," *J. Comput. Syst. Sci. 37*, 1, pp. 79–100,1988.

[15] M. Kearns, M. Littman and S. Singh. "Graphical Models for Game Theory," *Proceedings of UAI*, 2001.

[16] C. E. Lemke and J. T. Howson, Jr. "Equilibrium points of bimatrix games", *SIAM J. Appl. Math. 12*, pp. 413–423, 1964.

[17] R. Lipton and E. Markakis. "Nash Equilibria via Polynomial Equations," *Proceedings of LATIN*, 2004.

[18] R. Lipton, E. Markakis and A. Mehta. "Playing large games using simple strategies," *Proceedings of the ACM Conference on Electronic Commerce*, 2003.

[19] M. Littman, M. Kearns and S. Singh. "An Efficient Exact Algorithm for Single Connected Graphical Games," *Proceedings of NIPS*, 2002.

[20] J. Nash. "Noncooperative Games," *Annals of Mathematics, 54*, 289-295, 1951.

[21] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*, Princeton University Press, 1944.

[22] M.J. Osborne and A. Rubinstein. *A Course in Game Theory*, MIT Press (1994).

[23] C. H. Papadimitriou. "On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence," *J. Comput. Syst. Sci. 48*, 3, pp. 498–532, 1994.

[24] C. H. Papadimitriou and T. Roughgarden. "Computing equilibria in multi-player games," *Proceedings of SODA*, 2005.

[25] C. H. Papadimitriou. "Computing Correlated Equilibria in Multiplayer Games," *Proceedings of STOC*, 2005.

[26] R. Savani and B. von Stengel. "Exponentially many steps for finding a Nash equilibrium in a Bimatrix Game," *Proceedings of FOCS*, 2004.

[27] G. Schoenebeck and S. Vadhan. "The Computational Complexity of Nash Equilibria in Concisely Represented Games," *To appear in ACM EC*, 2006.