Nash Equilibria: Complexity, Symmetries, and Approximation

Constantinos Daskalakis*

September 2, 2009

Dedicated to Christos Papadimitriou, the eternal adolescent

Abstract

We survey recent joint work with Christos Papadimitriou and Paul Goldberg on the computational complexity of Nash equilibria. We show that finding a Nash equilibrium in normal form games is computationally intractable, but in an unusual way. It does belong to the class NP; but Nash's theorem, showing that a Nash equilibrium always exists, makes the possibility that it is also NP-complete rather unlikely. We show instead that the problem is as hard computationally as finding Brouwer fixed points, in a precise technical sense, giving rise to a new complexity class called PPAD. The existence of the Nash equilibrium was established via Brouwer's fixed-point theorem; hence, we provide a computational converse to Nash's theorem.

To alleviate the negative implications of this result for the predictive power of the Nash equilibrium, it seems natural to study the complexity of approximate equilibria: an efficient approximation scheme would imply that players could in principle come arbitrarily close to a Nash equilibrium given enough time. We review recent work on computing approximate equilibria and conclude by studying how symmetries may affect the structure and approximation of Nash equilibria. Nash showed that every symmetric game has a symmetric equilibrium. We complement this theorem with a rich set of structural results for a broader, and more interesting class of games with symmetries, called anonymous games.

1 Introduction

A recent convergence of Game Theory and Computer Science The advent of the Internet brought about a new emphasis in the design and analysis of computer systems. Traditionally, a computer system was evaluated in terms of its efficiency in the use of computational resources. The system's performance was determined by the design of its basic components along with the mechanisms by which these components were coordinated to work as a whole. With the growth of the Internet and the widely distributed nature of its design and operation, it became apparent that a system's performance is also intimately related to the selfish interests of the entities that run it and use it. And, in fact, this relation can be quite strong. If the designers of such systems ignore the potential competition or cooperation that may arise in the interaction of the systems' users or administrators, this could have destructive effects in the performance. Let us recall the recent two-hour outage in YouTube accesses throughout the globe, which evolved from a mere Border Gateway Protocol (BGP) table update—a result of censorship—in a network in Pakistan [48, 49]. . .

So, if computer scientists were to account for incentives in computer systems, what discipline should they appeal to for the tools that are needed? Naturally, game theory comes to mind. Since von Neumann's foundational contributions to the field in the 1920's, game theory has developed a rich set of models for

*CSAIL, MIT, costis@csail.mit.edu. This article was written while the author was a postdoctoral researcher in Microsoft Research, New England, and is based on research done at UC Berkeley and Microsoft Research.

abstracting mathematically situations of conflict, as well as *solution concepts*, tools for characterizing and, in certain cases, predicting the outcomes of these conflicts. However, when it comes to importing concepts from game theory to computer science, there is a twist: game theory evolved for the most part independently of computer science. So, it is a priori not clear what tools developed by game theorists are meaningful and relevant to the study of computer systems.

Naturally, the application of game theory to the study of computer systems has evolved along the following directions. First, it is important to obtain satisfactory models for the behavior of the entities, users, designers or administrators, that interact by running or using a computer system. Coming up with the right models is tightly related to the specifics of the users' interactions, as defined by the system design: the information shared by the users, the potential technological limitations in the users' actions, the opportunities or lack thereof to revise earlier choices, etc. See, e.g., Friedman and Shenker [18] for models of user behavior on the Internet.

Once the behavior of the users in a computer system is understood and modeled, we can define the "solution concept" that characterizes the steady state, static or dynamic, of the system, as this may arise from the interaction of its users. It is then natural to ask the following: How is the system's performance affected by the conflict of interests among its users? Or, posed differently, how far is the behavior of the system from the designer's objective as a result of the users' selfish behavior? An elegant quantification of this distance was given by Koutsoupias and Papadimitriou at the turn of the century [28]. Their suggestion was to consider the ratio between the worst case performance of the system in the presence of selfish behavior, and its optimal performance, if a central planner was capable of optimizing the performance with respect to each user's action, and imposing that action to the user. The resulting ratio, ingeniously coined *price of anarchy* by Papadimitriou [38], has been studied in the context of several applications, and has lead to important discoveries, e.g., in the context of traffic routing [43].

Of course, it is natural to prefer system designs with a small price of anarchy; that is, systems achieving their designer's goal, even when they are operated by rational agents who seek to optimize their own benefits. Inducing these agents to behave in line with the designer's objective, if at all possible, requires extra effort in the design of the system. Indeed, a good part of game theory, suggestively called *mechanism design*, is devoted to studying exactly what kinds of global properties can be achieved in a system operated by self-interested individuals. Nevertheless, to import this theory into the design of computer systems, an extra requirement should be met: the designs proposed by the theory should give rise to systems that are not only robust with respect to their users' incentives, but that are also computationally efficient. The resulting theory of *algorithmic mechanism design*, a fusion between mechanism design and the theory of computation, was suggested by Nisan and Ronen [37] in the late 1990's and has been the focus of much research over the past decade. The second part of [36] provides an excellent overview of this area.

When we import a solution concept from game theory to model the behavior of strategic agents in a system, it is natural to desire this concept to have good computational features; because if it does, then we could efficiently predict the behavior of the agents and use our predictions to inform the design and study of the system. But there is also another reason why the computational tractability of solution concepts is important, of more philosophical value. With these concepts we aspire to model the behavior of rational individuals; hence shouldn't the strategies they prescribe be efficiently computable? If instead, it required the world's fastest supercomputer a prohibitively large amount of computational resources to find these strategies, would it be reasonable to expect that rational agents would be able to find and adopt them?

This article is devoted to studying the computational complexity of solution concepts. Our goal is to understand how long it would take rational, self-interested individuals to converge to the behavior that a solution concept prescribes. We focus our attention on the concept of the *Nash equilibrium*, explained below, which is one of the most important, and arguably one of the most influential, solution concepts in game theory. We survey recent results, joint work with Paul Goldberg and Christos Papadimitriou [8],

proving that in general it is computationally hard to compute a Nash equilibrium; hence, we should not expect the Nash equilibrium to provide an accurate prediction of behavior in every setting. We alleviate this negative result by presenting broad settings where the Nash equilibrium has good computational features, reporting on joint work with Christos Papadimitriou [12, 13, 14, 15].

A mathematical theory of games We are all familiar with the *rock-paper-scissors* game: Two players with the same set of actions, "rock", "paper", and "scissors", play simultaneously against each other. Once they have chosen an action, their actions are compared, and each of them is rewarded or punished, or, if they choose the same action, the players tie. Figure 1 assigns numerical payoffs to the different outcomes of the game. In this tabular representation, the rows represent the actions of one player—call that player the *row player*—and the columns represent the actions of the other player—called the *column player*. Every entry of the table is then a pair of payoffs; the first is given to the row player and the second to the column player. For example, if the row player chooses "rock" and the column player chooses "scissors", then the row player wins a point and the column player looses a point.

	$rock$	$paper$	$scissors$
$rock$	$(0,0)$	$(-1,1)$	$(1,-1)$
$paper$	$(1,-1)$	$(0,0)$	$(-1,1)$
$scissors$	$(-1,1)$	$(1,-1)$	$(0,0)$

Figure 1: Rock-paper-scissors

So, what should we expect the players of a rock-paper-scissors game to do? If you are familiar with the game, then you should also be familiar with the strategy of randomizing uniformly among "rock", "paper", and "scissors". It stands to argue that any other strategy, which is not equally likely "rock", "paper", and "scissors", is not a good idea: if, e.g., your opponent believes that you are not likely to play "rock", then she can exploit this against you by playing "scissors", or, if you are also not likely to play "paper", by playing "rock". On the other hand, if both players use the uniform strategy, then none of them has an incentive to change her strategy.

It turns out that the rock-paper-scissors game has a structural property that makes it fairly simple for the players to figure out how to play: for any pair of actions of the two players, their payoff sum is zero. And, in such *zero-sum games*, a nice coincidence happens. To explain it, let the row player pick a randomized strategy that would optimize her payoff, if the column player punished her as much as he could given her strategy. And let the column player independently do the same calculation: find a strategy that would optimize his payoff, if the row player punished him as much as she could given his strategy. In his seminal 1928 paper [34], John von Neumann established that any such pair of strategies satisfies the following stability property: none of the players of the game can (unilaterally) improve her expected payoff by switching to a different strategy; that is, the pair of strategies is self-reinforcing, called an *equilibrium*.

Zero-sum games have very limited capabilities for modeling real-life competition [35]. What happens if a game is non zero-sum, or if it has more than two players? Is there still a stable collection of strategies for the players? As it turns out, regardless of the number of players and the players' payoffs, there always exists a self-sustaining collection of strategies—one in which no player can improve her payoff by unilaterally changing her strategy. But this does not follow from von Neumann's result. Rather, it was shown twenty years later by John Nash, in an ingenious paper [33] that greatly marked and inspired game theory; in honor of his result, we now call such a stable collection of strategies, a *Nash equilibrium*.

In fact, both von Neumann's and Nash's proofs are based on Brouwer's fixed-point theorem [27]. Still

they are fundamentally different: two decades after von Neumann's result [34], it became understood that the existence of an equilibrium in two-player zero-sum games is tantamount to linear-programming duality [7], and, as was established another three decades later, an equilibrium can be computed efficiently with linear programming [26]. On the other hand, despite much effort towards obtaining efficient algorithms for general games [29, 47, 42, 45], no efficient algorithm is known to date; and for most of the existing algorithms, exponential lower bounds have been established [44, 22]. So, *how hard is it to find a Nash equilibrium in general games?*

The ineffectiveness of the theory of NP-completeness The non-existence of efficient algorithms for finding Nash equilibria, after some fifty years of research on the subject, may be a strong indication that the problem is computationally intractable. Nonetheless, no satisfactory complexity lower bounds have been established either. For two-player games, there always exists a Nash equilibrium whose strategies have rational probability values [29]. If the support of these strategies was known, the equilibrium could be recovered by solving a linear program. Hence, the problem belongs to FNP, the class of function problems in NP. Alas, there are exponentially many possible pairs of supports. Is the problem then also FNP-complete? We do not know. But, we *do* know that the following, presumably harder, problems are FNP-complete: finding a Nash equilibrium with a certain payoff-sum for the players, if one exists; and finding two Nash equilibria, unless there is a single equilibrium [21, 6].

Indeed, NP draws much of its complexity from the possibility that a solution may not exist. Because a Nash equilibrium always exists, the problem of finding a single equilibrium is unlikely to be FNP-complete.[1] Rather, it belongs to the subclass of FNP containing problems that always have a solution. This class is called TFNP, for total function problems in NP, and contains such familiar problems as FACTORING. How hard then is NASH, the problem of finding a Nash equilibrium, with respect to the other problems in TFNP? To answer this question we have to take a closer look at the space of problems between FP and TFNP.

A look inside TFNP A useful and very elegant classification of the problems in TFNP was proposed by Papadimitriou in [39]. The idea is the following: If a problem is total, then there should be a proof showing that it always has a solution. In fact, if the problem is not known to be tractable, then this existence proof should be *non-constructive*; otherwise it would be easy to turn this proof into an efficient algorithm. We could then group the TFNP problems into classes, according to the non-constructive proof that is needed to establish their totality. It turns out that many of the hard problems in the fringes of P can be grouped according to the following existence proofs [39]:

- "If a directed graph has an unbalanced node—a node whose indegree is different from its outdegree—then it must have another unbalanced node." This parity argument on directed graphs gives rise to the class PPAD, which we study in this article as it captures the complexity of the Nash equilibrium.
- "If an undirected graph has an odd degree node, then it must have another odd degree node." This is the parity argument on undirected graphs, and it gives rise to a broader class, called PPA.
- "Every directed acyclic graph has a sink." The corresponding class is PLS, and it contains problems solvable by local search.
- "If a function maps n elements to $n-1$ elements, then there is a collision." This is the pigeonhole principle, and its class is called PPP.

But, wait! In what sense are these arguments non-constructive? Isn't it trivial to find an unbalanced node in a graph, a sink in a DAG, or a collision in a function? Indeed, it does depend on the way the graph or function is given. If the input, graph or function, is specified implicitly by a circuit, the problem may not be

[1] In fact, any attempt to show this will most likely have such unexpected implications as NP=co-NP [31].

as easy as it sounds. Consider, e.g., the following problem corresponding to the parity argument on directed graphs; we base our definition of PPAD on this problem, and the other classes are defined similarly [39]:

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

To understand the problem, let us think of P and S as specifying a directed graph with node set $\{0,1\}^n$ as follows: There is an edge from node u to node v iff $S(u) = v$ and $P(v) = u$; so, in particular, every node has in- and out-degree at most 1. Also, because $S(P(0^n)) \neq 0^n = P(S(0^n))$, the node 0^n has in-degree 0 and out-degree 1 and hence it is unbalanced. From the parity argument in directed graphs, there must be another unbalanced node, and we seek to find any unbalanced node different from 0^n.

The obvious approach to solve END OF THE LINE is to follow the path originating at 0^n until the sink node at the other end of this path is found. But this procedure may take exponential time, since there are 2^n nodes in the graph specified by S and P. To solve the problem faster, one should figure out a way to "telescope" the graph by looking at the details of the circuits S and P ...

The class PPAD Given the problem END OF THE LINE, we can define the class PPAD in terms of efficient reductions, in the same fashion that FNP can be defined via reductions from SATISFIABILITY, the problem of finding a satisfying assignment in a boolean formula.

Our notion of reduction is the following: A search problem $\mathcal{A}$ in TFNP is polynomial-time reducible to a problem $\mathcal{B}$ in TFNP, if there exists a pair of polynomial-time computable functions (f, g) such that, for every instance x of problem $\mathcal{A}$, $f(x)$ is an instance of problem $\mathcal{B}$, and for every solution y of the instance $f(x)$, $g(y, x)$ is a solution of the instance x. In this terminology, we define PPAD as the class of all problems in TFNP that are polynomial-time reducible to END OF THE LINE.

We believe that PPAD is a class of hard problems. But, since PPAD lies between FP and FNP, we cannot hope to show this without also showing that FP$\neq$FNP. In the absence of such proof, our reason to believe that PPAD is a hard class is the same reason of computational and mathematical experience that makes us believe that FNP is a hard class (even though our confidence is necessarily a little weaker for PPAD, since it lies inside FNP): PPAD contains many problems for which researchers have tried for decades to develop efficient algorithms; later in this survey, we introduce one such problem, called BROUWER, which regards the computation of approximate Brouwer fixed points. But, the problem END OF THE LINE is already a pretty convincingly hard problem: How could we hope to "telescope" exponentially large paths in every implicitly given graph?

Sperner's Lemma To grasp the power of PPAD a bit more, let us consider the following elegant result in Combinatorics, called *Sperner's Lemma*. To explain it, let us consider the unit square, subdivide it into smaller squares of size δ, and then divide each of these little squares into two right triangles, as shown in Figure 2—ignore colors, shadings, and arrows for a moment. We are allowed to color the vertices of this subdivision with three colors, red, yellow, and blue, in an arbitrary fashion, except our coloring must satisfy the following property:

(P_1): None of the vertices on the lower side of the square uses red, no vertex on the left side uses blue, and no vertex on the other two sides uses yellow. The resulting coloring may look like the one shown in Figure 2, ignoring the arrows and the shading of triangles.

Sperner's lemma asserts that, in any coloring satisfying property (P_1), there exists at least one small right triangle whose vertices have all three colors. Indeed, these triangles are shaded in Figure 2.

Imagine now that the colors of the vertices of the grid are not given explicitly. Instead, let us take $\delta = 2^{-n}$ and suppose that a circuit is given, which on input (x, y), $x, y \in \{0, 1, \ldots, 2^n\}$, outputs the color

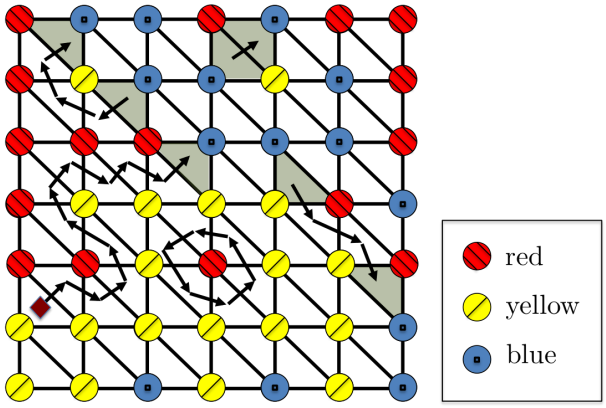

Figure 2: An illustration of Sperner's Lemma and its proof. The triangles correspond to the nodes of the END OF THE LINE graph, and the arrows to the edges; the source node T^* is marked by a diamond.

of the point $2^{-n} \cdot (x, y)$. How hard is it to find a trichromatic triangle now? We are going to argue that this problem belongs to PPAD. As a byproduct of this, we are also going to obtain a proof of Sperner's lemma. Our proof will construct an END OF THE LINE graph, whose solutions will correspond to the trichromatic triangles of the Sperner instance.

For simplicity, let us assume that there is only one change from yellow to red on the left side of the unit square, as in Figure 2. In the general case, we can expand the square by adding a vertical array of vertices to the left of the left side of the square, and color all of these vertices red, except for the bottom-most vertex which we color yellow; clearly, this addition doesn't introduce any new trichromatic triangles, since the edge of the square before the addition only had red and yellow vertices. Now the construction of the END OF THE LINE graph is done as follows. We identify the nodes of the graph with the right triangles of the subdivision, so that the node 0^n is identified with the triangle T^* that contains the unique, by assumption, segment on the left side of the unit square where the transition from yellow to red occurs (this triangle is marked in Figure 2 by a diamond). We then introduce an edge from node u to node v, if the corresponding right triangles T_u and T_v share a red-yellow edge, which goes from red to yellow clockwise in T_u. The edges are represented by arrows in Figure 2; the triangles with no incoming or outgoing arrows correspond to isolated nodes.

It is not hard to check that the constructed graph has in-degree and out-degree at most 1. Moreover, if T^* is not trichromatic, then it is the source of a path, and that path is guaranteed to have a sink, since it cannot intersect itself, and it cannot escape outside the square (notice that there is no red-yellow edge on the boundary that can be crossed outward). But, the only way a triangle can be a sink of this path is if the triangle is trichromatic! This establishes that there is at least one trichromatic triangle. There may of course be other trichromatic triangles, which would correspond to additional sources and sinks in the graph, as in Figure 2, so that the total number of trichromatic triangles is odd. To conclude the construction, observe that the circuit computing the coloring of the vertices of the subdivision can be used to construct the circuits S and P required to specify the instance of END OF THE LINE.

Back to Nash Equilibria We are now ready to come back to our original question about the complexity of NASH, the problem of computing a Nash equilibrium. For 2-player games, we argued that there always exists an equilibrium with rational probability values. But, as already pointed out by Nash in his original paper [33], there exist 3-player games with only irrational equilibria. In the absence of rational solutions, what does it really mean to compute a Nash equilibrium?

We treat irrational solutions by introducing a notion of approximation. To do this properly, we need a bit

of notation. Let us represent the players of a game by the numbers $1, \ldots, n$ and, for every $p \in \{1, \ldots, n\}$, let us denote by S_p the set of actions available to p. For every selection of actions by p and the other players of the game, p gets some *payoff*, or *utility*, value computed by a function $u_p : S_p \times \prod_{q \neq p} S_q \rightarrow [0, 1]$. Player p may in fact randomize by choosing a probability distribution x_p over her actions, so that $x_p(s_p) \geq 0$, for all $s_p \in S_p$, and $\sum_{s_p} x(s_p) = 1$. A randomized strategy of this sort is also called a *mixed strategy*.

In this notation, a Nash equilibrium is a collection of mixed strategies $x_1, \ldots, x_n$ such that, for all p, x_p maximizes the expected payoff to player p, if the other players choose an action from their mixed strategies independently at random. Equivalently, it must be that for every player p and for every pair of actions s_p, s'_p, with $x_p(s_p) > 0$:

$$\mathbb{E}[u_p(s_p\ ;\ s)] \geq \mathbb{E}[u_p(s'_p\ ;\ s)], \tag{1}$$

where, for the purpose of the expectation, s is chosen from the product measure defined by the mixed strategies $\{x_q\}_{q \neq p}$.

To accommodate approximate solutions in the Nash equilibrium problem, there are two obvious approaches. One is to allow in the output a collection of mixed strategies that is within distance ϵ, for some notion of distance, from an exact Nash equilibrium. This is a reasonable choice from a computational viewpoint. However, it is arguably unsatisfactory from a game-theoretic perspective. Indeed, the players of a game are interested in their payoff, and this is exactly what they aim to optimize with their strategy. The distance from an exact equilibrium is instead a global property of the strategies of all the players; and each individual player may not care about this distance, or may not even be in a position to measure it—if, e.g., she does not know the utility functions of the other players. . .

Motivated by this discussion, we opt for a different notion of approximation, defined in terms of the players' payoffs. We call a collection $x_1, \ldots, x_n$ of mixed strategies an *ϵ-Nash equilibrium* if, for every player p and for every pair of actions s_p, s'_p, with $x_p(s_p) > 0$:

$$\mathbb{E}[u_p(s_p\ ;\ s)] \geq \mathbb{E}[u_p(s'_p\ ;\ s)] - \epsilon,$$

where s is chosen from the product measure defined by the mixed strategies $\{x_q\}_{q \neq p}$. That is, we allow in the support of the mixed strategy of player p actions that approximately optimize p's expected payoff. If ϵ is sufficiently small, e.g., smaller than the quantization of the currency used by the players, or the cost of changing one's strategy, it may be reasonable to assume that approximate payoff optimality is good enough for the players. Note also that it is easy for a player to check the approximate optimality of the actions used by her mixed strategy. So the notion of an ϵ-Nash equilibrium is much more appealing as a notion of approximate Nash equilibrium; and, in fact, this is the notion most commonly used [45].

Given that there always exists a rational ϵ-Nash equilibrium in a game, the following is a computationally meaningful definition of NASH, the problem of computing a Nash equilibrium.

NASH: Given a game $\mathcal{G}$ and an approximation requirement ϵ, find an ϵ-Nash equilibrium of $\mathcal{G}$.

The main result in this survey is the following result from joint work with Goldberg and Papadimitriou [8].

Theorem 1 ([8]) NASH *is* PPAD-complete.

We provide the proof of Theorem 1 in Sections 2 and 3. The proof has two parts: We show first that the problem of computing (approximate) Brouwer fixed points is PPAD-complete (Theorem 2, Section 2). Given this, we immediately establish that NASH is in PPAD, since the existence of a Nash equilibrium was established by Nash via Brouwer's fixed-point theorem [33]. We spend most of Section 3 to show the opposite reduction, that computing (approximate) Brouwer fixed points can be reduced to NASH; and this

establishes the PPAD-hardness of NASH. Our original proof of this result, reported in [8, 11], showed the hardness of NASH for games with at least 3 players. Soon after our proof became available, Chen and Deng provided a clever improvement of our techniques, extending the result to 2-player games [4].

Discussion of the main result Theorem 1 implies that NASH is at least as hard as any problem in PPAD. So, unless PPAD=FP, there is no efficient algorithm for solving NASH. If this is the case, there exist games in which it would take the players excruciatingly long to find and adopt the Nash equilibrium strategies. In such a game, it does not seem reasonable to expect the players to play a Nash equilibrium. Hence, Theorem 1 can be seen as a computational critique to the accuracy of the Nash equilibrium as a framework for predicting behavior in all games. . .

Another interesting aspect of Theorem 1 is that it identifies exactly the non-constructive argument needed to obtain the existence of a Nash equilibrium. That an algorithm for END OF THE LINE is sufficient for finding ϵ-Nash equilibria was, in fact, implicit in the Nash equilibrium algorithms suggested since the 1960's [29, 42, 47, 45]. Theorem 1 formalizes this intuition in a computationally meaningful manner and more importantly establishes that NASH is at least as hard as the generic END OF THE LINE problem. Moreover, since the problem BROUWER of computing approximate Brouwer fixed points is also PPAD-complete (see Section 2), Theorem 1 provides a computational converse to Nash's result.

As we already argued, in view of Theorem 1, it is hard to believe that the Nash equilibrium predicts accurately how players behave in a situation of conflict. Does that imply that we should dismiss the concept of the Nash equilibrium altogether? We think that there is still a consolation: It could be that players never play an exact Nash equilibrium, but as time progresses they *do* approach an equilibrium by, say, becoming wiser as to how to play. Of course, since NASH is PPAD-complete, we know that, unless PPAD=FP, there is no efficient algorithm for computing an ϵ-Nash equilibrium, if ϵ scales inverse exponentially in the size of the game. Hence, it is unlikely that the players will approximate the equilibrium to within an exponential precision in polynomial time. In fact, this hardness result has been extended to ϵ's that scale inverse polynomially in the size of the game [5]; so polynomial precision seems unlikely too. Nevertheless, the following possibility has not been yet eliminated: an algorithm for finding an ϵ-Nash equilibrium in time $n^{f(1/\epsilon)}$, where n is the size of the game and $f(1/\epsilon)$ is some function of ϵ. Such an algorithm would be consistent with the possibility that the players of a game can discover an ϵ-Nash equilibrium, for any desired approximation ϵ, as long as they play for a long enough period of time, depending super-polynomially in ϵ, but polynomially in the details of the game. In Section 4, we discuss recent results on this important open problem, and provide efficient algorithms for a broad and appealing class of multi-player games with symmetries, called *anonymous games*.

2 The Complexity of Brouwer Fixed Points and PPAD

We already mentioned that the existence of a Nash equilibrium was established by Nash with an application of Brouwer's fixed-point theorem. Even if you're not familiar with the statement of this theorem, you are most probably familiar with its applications: Take two identical sheets of graph paper with coordinate systems on them, lay one flat on the table and crumple up (without ripping or tearing) the other one and place it any fashion you want on top of the first so that the crumpled paper does not reach outside the flat one. There will then be at least one point of the crumpled sheet that lies exactly on top of its corresponding point (i.e. the point with the same coordinates) of the flat sheet. The magic that guarantees the existence of this point is topological in nature, and this is precisely what Brouwer's fixed-point theorem captures. The statement formally is that any continuous map from a *compact* (that is, closed and bounded) and *convex* (that is, without holes) subset of the Euclidean space into itself always has a *fixed point*.

Brouwer's theorem suggests the following interesting problem: Given a continuous function F from some compact and convex subset of the Euclidean space to itself, find a fixed point. Of course, to make this problem computationally meaningful we need to address the following: How do we represent a compact and convex subset of the Euclidean space? How do we specify the continuous map from this set to itself? And how do we deal with the possibility of irrational fixed points?

For simplicity, let us fix the compact and convex set to be the unit hypercube $[0,1]^m$. We could allow the set to be any convex polytope (given as the bounded intersection of a finite set of half-spaces, or as the convex hull of a finite set of points in $\mathbb{R}^m$). We could even allow more general domains as long as these can be described succinctly, e.g., spherical or ellipsoidal domains. Usually, the problem in a more general domain can be reduced to the hypercube by shrinking the domain, translating it so that it lies inside the hypercube, and extending the function to the whole cube so that no new fixed points are introduced.

We also assume that the function F is given by an efficient *algorithm* Π_F which, for each point x of the cube written in binary, computes $F(x)$. We assume that F obeys a Lipschitz condition:

$$d(F(x_1), F(x_2)) \leq K \cdot d(x_1, x_2), \ \ \forall x_1, x_2 \in [0,1]^m \tag{2}$$

where $d(\cdot,\cdot)$ is the Euclidean distance, and K is the *Lipschitz constant* of F. This benign well-behavedness condition ensures that approximate fixed points can be localized by examining the value $F(x)$ when x ranges over a discretized grid over the domain. Hence, we can deal with irrational solutions in a similar maneuver as with NASH, by only seeking points which are approximately fixed by F. In fact, we have the following strong guarantee: for any ϵ, there is an ϵ-approximate fixed point, that is, a point x such that $d(F(x), x) \leq \epsilon$, whose coordinates are integer multiples of 2^{-d}, where 2^d is linear in K, $1/\epsilon$ and the dimension m; in the absence of a Lipschitz constant K, there would be no such guarantee and the problem of computing fixed points would become much more difficult. Similarly to the case of Nash equilibria, we can define an alternative (stronger) notion of approximate Brouwer fixed points, by insisting that these are in the proximity of exact fixed points. Of course, which of these two notions of approximation is more appropriate depends on the underlying application. Since our own motivation for considering Brouwer fixed points is their application to Nash equilibria, we opt for the first definition, which as we will see in Section 3 relates to ϵ-Nash equilibria. A treatment of the stronger notion of approximation can be found in [17].

Putting everything together, the problem BROUWER of computing approximate fixed points is defined as follows.

BROUWER

Input: An efficient algorithm Π_F for the evaluation of a function $F : [0,1]^m \to [0,1]^m$; a constant K such that F satisfies Condition (2); and the desired accuracy ϵ.

Output: A point x such that $d(F(x), x) \leq \epsilon$.

In the following subsections, we will argue that PPAD captures precisely the complexity of BROUWER. Formally,

Theorem 2 BROUWER *is* PPAD*-complete.*

2.1 From Fixed Points to Paths: BROUWER $\in$ PPAD

We argue that BROUWER is in PPAD, by constructing an END OF THE LINE graph associated with a BROUWER instance. To define the graph, we construct a mesh of tiny simplices over the domain, and we identify each of these simplices with a vertex of the graph. We then define edges between simplices sharing a facet with respect to a coloring of the vertices of the mesh. The vertices get colored according to

the direction in which F displaces them. We argue that if a simplex's vertices get all possible colors, then F is trying to shift these points in conflicting directions, and we must be close to an approximate fixed point. We elaborate on this in the next few paragraphs, focusing on the 2-dimensional case.

Triangulation First, we subdivide the unit square into small squares of size determined by ϵ and K, and then divide each little square into two right triangles (as shown in Figure 2, ignoring colors, shading, and arrows). In the m-dimensional case, we subdivide the m-dimensional cube into m-dimensional cubelets (their size depends now also on the number of dimensions m), and we subdivide each cubelet into m-simplices. There are many ways to do this. Here is one: Suppose that the cubelet is $[0, \lambda]^m$. For a permutation $\sigma = (i_1, ..., i_m)$ of $(1, ..., m)$, let h_σ be the subset of the cubelet containing those points satisfying $0 \leq x_{i_1} \leq x_{i_2} \leq \ldots \leq x_{i_m} \leq \lambda$. Then the $m!$ simplices h_σ, as σ ranges over all permutations of $(1, \ldots, m)$ provide the subdivision.

Coloring We color each vertex x of the simplicization of $[0, 1]^m$ by one of $m + 1$ colors depending on the *direction* in which F maps x. In two dimensions, this can be taken to be the angle between vector $F(x) - x$ and the horizontal. Specifically, we can color a vertex red if the direction lies between 0 and -135 degrees, blue if it ranges between 90 and 225 degrees, and yellow otherwise, as shown in Figure 3. (If the direction is 0 degrees, we allow either yellow or red; similarly for the other two borderline cases.) Using this coloring convention the vertices can get colored in such a way that the following property is satisfied:

(P_2): None of the vertices on the lower side of the square uses red, no vertex on the left side uses blue, and no vertex on the other two sides uses yellow.

The resulting coloring could look like the one shown in Figure 2, ignoring arrows and shadings of triangles. The multidimensional analog of this coloring is more involved, but similar in spirit. We partition the set of directions into colors, $0, 1, \ldots, m$, in such a way that the coloring of the vertices satisfies the following property:

(P_m): For all $i \in \{1, \ldots, m\}$, none of the vertices on the face $x_i = 0$ of the hypercube uses color i; moreover, color 0 is not used by any vertex on the face $x_i = 1$, for all $i \in \{1, \ldots, m\}$.

Roughly, color 0 is identified with the directions corresponding to the positive quadrant of the m-dimensional space. The remaining directions are partitioned into the colors $1, \ldots, m$ evenly so that property P_m is realized.

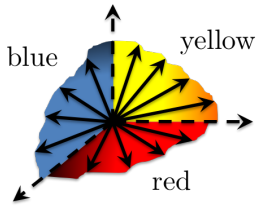

Figure 3: The colors assigned to the different directions of $F(x) - x$. There is a transition from red to yellow at 0 degrees, from yellow to blue at 90 degrees, and from blue to red at 225 degrees.

Sperner, Again Since our coloring satisfies property (P_m), it follows from the m-dimensional analog of Sperner's lemma that there exists a simplex whose vertices have all $m + 1$ colors. Because the size of the simplices was chosen to be sufficiently small, and our colors span the space of directions evenly, it can be

shown that any vertex of a trichromatic triangle will be an approximate fixed point. Intuitively, since F satisfies the Lipschitz condition given in (2), it cannot fluctuate too fast; hence, the only way that there can be $m+1$ points close to each other in distance, which are mapped in $m+1$ different directions, is if they are all approximately fixed. Hence, solving BROUWER reduces to finding a panchromatic simplex in an instance of Sperner's lemma. We already argued that the 2-dimensional version of this problem is in PPAD, and similar arguments establish that the m-dimensional version is also in PPAD. This concludes the proof that BROUWER is in PPAD.

2.2 From Paths to Fixed Points: BROUWER is PPAD-complete

We need to show how to encode an END OF THE LINE graph G in terms of a continuous, easy-to-compute Brouwer function F, a very different-looking mathematical object. The encoding is unfortunately rather complicated, but is key to the PPAD-completeness result...

We proceed by, first, using the 3-dimensional unit cube as the domain of the function F. Next, the behavior of F shall be defined in terms of its behavior on a (very fine) rectilinear mesh of "grid points" in the cube. Thus, each grid point lies at the center of a tiny "cubelet", and the behavior of F away from the centers of the cubelets shall be gotten by interpolation with the closest grid points. Each grid point x shall receive one of 4 "colors" $\{0,1,2,3\}$, that represent the value of the 3-dimensional displacement vector $F(x)-x$. The 4 possible vectors can be chosen to point away from each other such that $F(x)-x$ can only be approximately zero in the vicinity of all 4 colors. We choose the following direction for our colors: $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, and $(-1,-1,-1)$.

We are now ready to fit G itself into the above framework. Our construction is illustrated in Figure 4. Each of the 2^n nodes of G corresponds to a small segment aligned along an edge of the cube (in Figure 4, the segments $v_1 \to v_1'$ and $u_1 \to u_1'$ correspond respectively to the nodes v and u of G; we use locations that are easy to compute from the identity of a vertex of G). Each edge of G corresponds to a sequence of connected segments in the interior of the cube (Figure 4 shows the path corresponding to the directed edge $u \to v$: it starts at node u_1' and ends at node v_1). It is important that, whether such a path goes through a grid point, and which direction it is going, can be computed locally at the point using only the circuits P and S of the END OF THE LINE instance. Then these paths are the basis of defining an efficiently computable coloring of the grid points, such that the fixed points of the resulting function F correspond to the unbalanced nodes of G.

Here is the gist of the construction: Within a certain radius R from the line corresponding to the paths and cycles of G, the grid points are moved by F towards non-decreasing values of the coordinates x, y, and z, in a complicated manner such that the only points moved by F towards a direction that simultaneously increases all three coordinates lie within an even smaller radius r from the line; let us call these latter points *increasing*. Then the points lying at distance larger than R from the line are moved by F towards the direction $(-1,-1,-1)$; let us call these points *decreasing*. Overall the construction of F guarantees that an approximate fixed point may only occur if an increasing point comes close to a decreasing point. But there is a layer of thickness $R-r$ protecting the increasing points from the decreasing points, so that the only areas of the cube where the increasing points are exposed to the decreasing ones are near the endpoints of segments that correspond to the unbalanced nodes of the END OF THE LINE graph. This completes the sketch of the PPAD-completeness of BROUWER.

3 The Complexity of Nash Equilibria

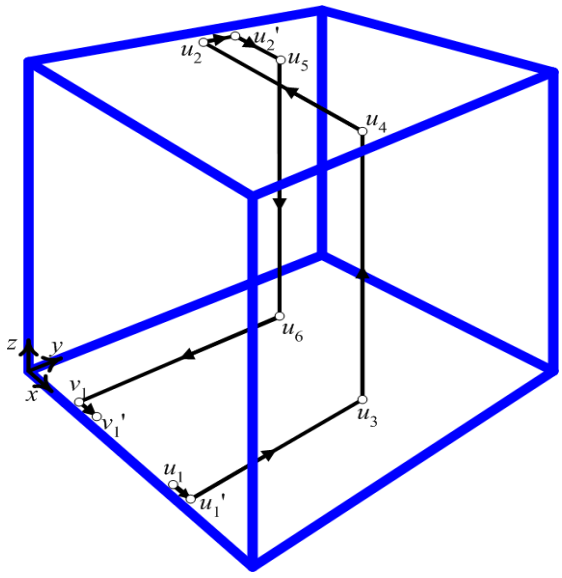

Figure 4: Embedding the END OF THE LINE graph in a cube. The embedding is used to define a continuous function F, whose approximate fixed points correspond to the unbalanced nodes of the END OF THE LINE graph.

We come back to the proof of Theorem 1. To show that NASH is in PPAD we appeal to Nash's proof [33], which essentially constitutes a reduction from NASH to BROUWER. For the hardness result, we need to show a computational converse to Nash's theorem, reducing the computation of fixed points of arbitrary Brouwer functions to the computation of Nash equilibria. We discuss these reductions in the following subsections.

3.1 From Nash Equilibria to Brouwer Fixed Points: NASH $\in$ PPAD

Here is a high level description of Nash's argument: Suppose that the players of a game have chosen some (mixed) strategies. Unless these already constitute a Nash equilibrium, some of the players will be unsatisfied, and will wish to change to some other strategies. This suggests that one can construct a "preference function" F_N from the set of players' strategies to itself, that indicates the changes that will be made by any unsatisfied player. A *fixed point* of such a function is a point that is mapped to itself—a Nash equilibrium. And Brouwer's fixed-point theorem, explained above, guarantees that such a fixed point exists.

Given Nash's proof, in order to conclude our reduction from NASH to BROUWER, we need to show the following: first, for every input $x = (x_1, \ldots, x_n)$, the value $F_N(x)$ should be efficiently computable given the description of the game; second, F_N should satisfy the Lipschitz condition (2) for some well-behaved value of K; finally, the approximate fixed points of F_N should correspond to approximate Nash equilibria. It turns out that Nash's function satisfies all these properties (see [8] for details), and this finishes our proof that NASH reduces to BROUWER, and hence also to PPAD.

We conclude by illustrating Nash's function for the *penalty shot* game shown in Figure 5. In this game, there are two players, the goalkeeper and the penalty kicker, they both have the same actions, 'left' and 'right', and if they choose the same action the goalie wins, while if they choose opposite actions the penalty kicker wins. Figure 6 shows Nash's function for this game. The horizontal axis represents the probability by which the penalty kicker kicks right, and the vertical axis the probability by which the goalkeeper dives left, while the arrows correspond to the direction and magnitude of $F_N(x) - x$. For example, in the bottom right corner of the square, which corresponds to both of the players playing 'right', $F_N(x) - x$ is large and pointing to the left. This indicates the fact that in that corner the goalkeeper is happy for his strategy, while

the penalty kicker experiences a large regret and desires to switch to her 'left' strategy. The unique fixed point of the function is the point $(1/2, 1/2)$, which corresponds to the unique Nash equilibrium of the game. The colors used in Figure 6 respect Figure 3, but our palette here is continuous.

	kick left	*kick right*
dive left	$(1, -1)$	$(-1, 1)$
dive right	$(-1, 1)$	$(1, -1)$

Figure 5: The *penalty shot* game. The rows correspond to the actions of the goalkeeper and the columns to the actions of the penalty kicker. The game has a unique Nash equilibrium in which both the goalkeeper and the penalty kicker choose 'left' or 'right' uniformly at random.

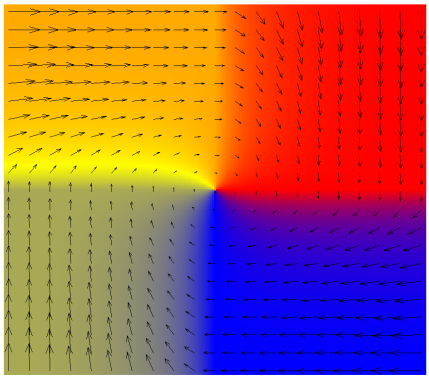

Figure 6: An illustration of Nash's function F_N for the penalty shot game. The horizontal axis represents the probability by which the penalty kicker kicks right, and the vertical axis the probability by which the goalkeeper dives left.

3.2 From Brouwer Fixed Points to Nash Equilibria: NASH is PPAD-complete

Our plan is to show that NASH is PPAD-complete by reducing BROUWER to it. To succeed in this quest we need to show how to encode fixed points of arbitrary functions into the Nash equilibria of games, the converse of Nash's encoding. In fact, it is sufficient to show this encoding for the class of PPAD-complete functions identified in Section 2.2, which are computable by circuits built up using a small repertoire of boolean and arithmetic operations, and comparisons. These circuits can be written down as a "data flow graph", with one of these operators at each node. We transform this into a game whose Nash equilibria correspond to (approximate) fixed points of the Brouwer function, by introducing players for every node on this data flow graph. The basic components of our construction are sketched below.

The High Level Structure of the Game We introduce three players X_1, X_2 and X_3 whose strategies are meant to identify a point in the cube. Here is how: we give each of these players two strategies 'stop' and 'go', so that, jointly, their probabilities x_1, x_2 and x_3 of choosing 'go' correspond to a point in $[0, 1]^3$. The way these players decide on their mixed strategies depends on their payoff functions, which we haven't fixed yet. Now we would like to have a game that "reads in" x_1, x_2 and x_3 and simulates the circuit computing F so that, in any Nash equilibrium, three players Y_1, Y_2 and Y_3 of this game represent $F(x)$ with their 'go'

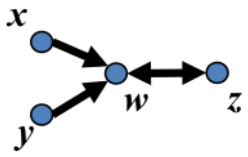

Figure 7: The players of the multiplication game. The graph shows which players affect other players' payoffs.

probabilities. If we had this game, we would be essentially done: we could give payoffs to the players X_1, X_2 and X_3 to ensure that in any Nash equilibrium, their 'go' probabilities agree with those of players Y_1, Y_2 and Y_3. Then, in any Nash equilibrium, these probabilities would have to be a fixed point of F!

Simulating Circuit Computations with Games So all we need is a game that simulates the circuit that computes F at a given input point x_1, x_2 and x_3. It is natural to construct this game by introducing a player for each intermediate node of the circuit, and define the payoffs of these players in a gate-by-gate fashion to induce them to implement the computations that the corresponding nodes in the circuit would do. We can actually assume that all the (non-binary) values computed by the nodes of the circuit lie in $[0, 1]$, so that our players can just have two actions, 'stop' and 'go', and "hold" the values of the circuit in their 'go' probabilities.

Let us now explore how we could go about simulating a gate of the circuit, e.g., a gate that multiplies two values. We need to choose the payoffs of three players X, Y and Z, so that, in any Nash equilibrium, the 'go' probability z of player Z is the product of the 'go' probabilities x and y of the players X and Y. We do this indirectly by introducing another player W who mediates between players X, Y and Z. The resulting "multiplication gadget" is shown in Figure 7, where the directed edges show the direct dependencies among the players' payoffs. Elsewhere in the game, player Z may "input" his value into other related gadgets.

Here is how we define the payoffs of the players X, Y, W and Z to induce them to implement multiplication: We pay player W an expected payoff of $\$x \cdot y$ for choosing 'stop' and $\$z$ for choosing 'go'. We also pay Z to play the opposite from player W. It is not hard to check that in any Nash equilibrium of the game thus defined, it must be the case that $z = x \cdot y$. (For example, if $z > x \cdot y$, then W would prefer strategy 'go', and therefore Z would prefer 'stop', which would make $z = 0$, and would violate the assumption $z > x \cdot y$.) Hence, the rules of the game induce the players to implement multiplication in the choice of their mixed strategies.

By choosing different sets of payoffs, we could ensure that $z = x + y$ or $z = \frac{1}{2}x$. We could also simulate boolean operators, but we must be careful when we use them: we should guarantee that the players corresponding to the inputs of these operators play 'go' with probability either 0 or 1, in every Nash equilibrium. If the inputs to a boolean operator are not binary, then its output could also be non-binary and this could affect other gates and eventually destroy the whole computation. So, to use our boolean gadgets with confidence, we need to look into the details of the functions arising in the proof of Section 2.2, understand what the inputs to the boolean operators are computing, and, hopefully, argue that in any Nash equilibrium these are binary...

The Brittle Comparator Problem It turns out that, for the functions of Section 2.2, all the nodes of the circuit that hold binary values are the results of comparisons. And there is a problem in our simulation of comparisons by games: *our comparator gadget*, whose purpose is to compare its inputs and output a binary signal according to the outcome of the comparison, is "brittle" in that if the inputs are equal then it outputs *anything*. This is inherent, because one can show that, if a non-brittle comparator gadget existed, then we

could construct a game that has no Nash equilibria, contradicting Nash's theorem. With brittle comparators, our computation of F is faulty on inputs that cause the circuit to make a comparison of equal values,[2] and this poses a serious challenge in our construction. We solve the problem by computing the Brouwer function at a grid of many points near the point of interest, and averaging the results. By doing this we can show that most of the circuit computations will not run into the brittle comparator problem, so our computation will be approximate, but *robust*. Hence, our construction *approximately* works, and the three special players of the game play an approximate fixed point at equilibrium.

Reducing the number of players The game thus constructed has many players (the number depends mainly on how complicated the circuit for computing the function F was), and two strategies for each player. This presents a problem: To represent such a game with n players we need $n2^n$ numbers—the utility of each player for each of the 2^n strategy choices of the n players. But our game has a special structure (called a *graphical game,* see [25]): The players are vertices of a graph (essentially the data flow graph of F), and the utility of each player depends only on the actions of its neighbors.

The final step in the reduction is to simulate this game by a three-player game—this establishes that NASH is PPAD-complete even in the case of three players. This is accomplished as follows: We color the players (nodes of the graph) by three colors, say red, blue, and yellow, so that no two players who play together, or two players who are involved in a game with the same third player, have the same color (it takes some tweaking and argument to make sure the nodes can be so colored). The idea is now to have three "lawyers", the red lawyer, the blue lawyer, and the yellow lawyer, each represent all nodes with their color, in a game involving only the lawyers. A lawyer representing m nodes has $2m$ actions, and his mixed strategy (a probability distribution over the $2m$ actions) can be used to encode the simpler stop/go strategies of the m nodes. Since no two adjacent nodes are colored the same color, the lawyers can represent their nodes without a "conflict of interest," and so a mixed Nash equilibrium of the lawyers' game will correspond to a mixed Nash equilibrium of the original graphical game.

But there is a problem: We need each of the "lawyers" to allocate equal amounts of probability to their customers; however, with the construction so far, it may be best for a lawyer to allocate more probability mass to his more "lucrative" customers. We take care of this last difficulty by having the lawyers play, on the side and for high stakes, a generalization of the rock-paper-scissors game of Figure 1, one that forces them to balance the probability mass allocated to the nodes of the graph. This completes the reduction from graphical games to three-player games, and the proof.

4 Symmetries and the Approximation of Nash Equilibria

In view of the hardness result for computing a Nash equilibrium described above, it is important to study if there is an efficient algorithm for ϵ-Nash equilibria when ϵ is fixed.[3] Such an algorithm would be consistent with the following justification of the Nash equilibrium as a framework for behavior prediction: even when the players of a game have trouble finding an exact equilibrium, it could be that given enough time they converge to an ϵ-Nash equilibrium. If ϵ is sufficiently small, say smaller than the cost of updating one's strategy, or the quantization of the currency used by the players, an ϵ-Nash equilibrium could be a stable state of the game.

Due to the relevance of approximate equilibria for the predictive power of the Nash equilibrium, their complexity has been extensively researched over the past few years. Most of the studies focus on two-player

[2] This happens for points of the cube that fall midway between 2 grid points, or more generally on the boundary of two simplices used for interpolating F from its values at the grid points—see Section 2.2.

[3] As mentioned earlier, going beyond constant ϵ's is probably impossible, since the problem is already PPAD-complete for ϵ's that scale inverse polynomially with the size of the game [5].

games [24, 15, 9, 10, 2], but even there the best approximation known to date is $\epsilon = 0.34$ [46].[4] On the other hand, if we are willing to give up efficient computation, there exists an algorithm for games with a constant number of players running in time $m^{O(\log m/\epsilon^2)}$, where m is the number of strategies per player [30]. The algorithm is based on the following interesting observation: for any ϵ, there exists an ϵ-Nash equilibrium in which the players randomize uniformly over a multiset of their strategies of size $O(\log m/\epsilon^2)$. Hence, we can exhaustively search over all collections of multisets of size $O(\log m/\epsilon^2)$ (one multiset for every player of the game) and then check if the uniform distributions over these multisets constitute an ϵ-Nash equilibrium. This last check can be performed efficiently, and we only pay a super-exponential running time because of the exhaustive search.

So, is it possible to remove the factor of $\log m$ from the exponent of the running time? This would give a polynomial-time approximation scheme (PTAS) for ϵ-Nash equilibria. While (as of the time of writing) we don't know the answer to this question, we do know that we cannot hope to get an efficient algorithm that is also *oblivious*. To explain what we mean by "oblivious", note that the algorithm we just described has a very special obliviousness property: no matter what the input game is, the algorithm searches over a fixed set of candidate solutions—all collections of uniform distributions on multisets of size $O(\log m/\epsilon^2)$—and it only looks at the game to verify whether a candidate solution is indeed an ϵ-Nash equilibrium. In recent joint work with Papadimitriou [15], we show that any oblivious algorithm of this kind needs to run in time $m^{O_\epsilon(\log m)}$. Hence, the algorithm presented above is essentially tight among oblivious algorithms and the $\log m$ in the exponent is inherent. Whether a non-oblivious PTAS for computing equilibria exists still remains one of the most important open problems in the area of equilibrium computation.

Games with Symmetries In the remaining of this section, we look at the problem from a different perspective, considering games with symmetries, and study how these affect the structure and approximation of Nash equilibria. Symmetries were actually already studied by Nash, in his original paper [33], where an interesting connection between the symmetries of a game and the structure of its Nash equilibria was established.

To explain Nash's result, let us consider a game with n players and the same set of strategies for all players. Let us also assume that the payoff of each player is symmetric with respect to the actions of the other players; in other words, every player only cares in her payoff about her own strategy and the number of players choosing every strategy, but she does not care about the identity of these players. If a game has this structure, and the players have identical payoff functions, the game is called *symmetric*. A trivial example is the rock-paper-scissors game of Figure 1. The game is symmetric because there are only two players, and their payoff functions are identical. For a more elaborate example, consider a group of drivers driving from Boston to New York on a weekend: they can all select from the same set of available routes, and their travel time will depend on their own choice of route and the number, but not the identity, of the other drivers who choose intersecting routes. Nash's result for symmetric games is the following.

Theorem 3 ([33]) *In every symmetric game, there exists an equilibrium in which all players use the same mixed strategy.*

This structural result was exploited by Papadimitriou and Roughgarden [40] to obtain an efficient algorithm for Nash equilibria in multiplayer symmetric games. Their algorithm is efficient as long as the number of strategies is small, up to $m = O(\log n/\log\log n)$ strategies, where n is the number of players. And the idea is quite simple:

1. first, we can guess the support of a symmetric equilibrium—there are 2^m possibilities and m was assumed to be smaller than $O(\log n)$;

[4] Recall that all payoff values were assumed to lie in $[0, 1]$, so that the additive approximation of 0.34 is meaningful.

2. now, we can write down the Nash equilibrium conditions: the expected payoff from every action in the support of the Nash equilibrium should be larger than or equal to the expected payoff from every other action (see Condition (1) in Section 1);
3. this results in a system of m polynomial equations and inequalities of degree $n-1$ in m variables; and it follows from the existential theory of the reals [41] that such a system can be solved to within any desired accuracy ϵ in time polynomial in n^m, $\log(1/\epsilon)$, and the input size.

Observe that, if the game is symmetric and the number of strategies is $m = O(\log n / \log\log n)$, then its description complexity is $n^{O(m)}$. Hence, the above algorithm runs in polynomial time in the input size!

But what if our symmetric game has a large number of strategies and a small number of players? Can we still solve the game efficiently in this case? It turns out that, here, there is not much we can do. From a symmetrization construction of von Neumann [3]—and another more efficient (computationally) construction by Gale, Kuhn and Tucker [19], it follows that two-player games can be reduced to symmetric two-player games. And, this quickly implies that two-player symmetric games are PPAD-complete. So we have no luck to approximate these games unless we find a PTAS for general two-player games...

And, going back to multi-player games, what if we consider games with fewer symmetries than symmetric games? Is the theory of symmetric equilibria applicable in other classes of games? We study these questions by considering a broad and important generalization of symmetric games, called *anonymous games*. These games are the focus of the remaining of this section.

Anonymous Games... are like symmetric games, in that all the players have the same set of strategies, and their payoff functions do not depend on the identities of the other players. But we now drop the assumption that their payoff functions are identical. As a result, anonymous games are broader and far more interesting for applications than symmetric games. Consider, e.g., a road network. It makes sense to assume that the drivers do not care about the identities of the other drivers.[5] On the other hand, it is highly unlikely that all the drivers have the same payoff function: they probably all have their own source-destination pairs, and their own way to evaluate their driving experience. So, the anonymous model fits much better for capturing the interactions of players in this game than the symmetric model. Anonymous games have been used to model several other situations, including auctions and the study of social phenomena; we refer the interested reader to [32, 1, 23] for a collection of papers on the subject by economists.

So, how much of the theory of symmetric equilibria applies to the anonymous setting? And, can we find equilibria efficiently in these games? We give answers to these questions by providing a sequence of polynomial-time approximation schemes for multi-player anonymous games with two strategies per player.[6] Every algorithm in this sequence is based on a progressively refined understanding of the structure of equilibria in these games. Observe that in the opposite setting of parameters, when the number of players is small and the number of strategies is large, we cannot hope to provide analogous results unless we also provide a PTAS for two-player games; since, in particular, two-player games are anonymous and fall into this case.

Before proceeding with our study, let us get our notation right. Take an anonymous game with n players, $1, \ldots, n$, and two strategies, 0 and 1, available to each player. To define the payoff of each player i we need to give two functions, $u_i^0, u_i^1 : \{0, \ldots, n-1\} \to [0,1]$, so that, for all $k \leq n-1$, $u_i^0(k)$ and $u_i^1(k)$ represent the payoff to player i, if she chooses 0 and 1 respectively, and k of the other players choose strategy 1. Now note that the mixed strategy of each player i is just a real number $q_i \in [0,1]$, corresponding to the probability by which the player plays strategy 1. For convenience, we can define an indicator random variable X_i, with expectation $\mathbb{E}(X_i) = q_i$, to represent this mixed strategy. And because the players randomize independently

[5] If you want to be more elaborate, the drivers may be partitioned into a small number of types, depending on their vehicle type and driving style, and then each driver would care about the type to which every other driver belongs, and the route that she chooses, but not her identity. All our results hold with appropriate modifications if there is a constant number of player types.

[6] Most of our results extend to a constant number of strategies per player.

from each other, we can assume that the indicators $X_1, \ldots, X_n$ representing the players' mixed strategies are mutually independent.

Since we are interested in approximating Nash equilibria, it makes sense to study, as a first step, the impact on a player's payoff that results from replacing a set of mixed strategies $X_1, \ldots, X_n$ by another set $Y_1, \ldots, Y_n$. For a fixed player i and action $s_i \in \{0, 1\}$, it is not hard to obtain the following bound:

$$\left|\mathbb{E}[u_i(s_i\ ;\ k)] - \mathbb{E}'[u_i(s_i\ ;\ k)]\right| \leq \left\| \sum_{j \neq i} X_j - \sum_{j \neq i} Y_j \right\|_{\mathrm{TV}}, \tag{3}$$

where, for the purposes of computing the expectation $\mathbb{E}$, k is drawn from $\sum_{j \neq i} X_j$, while for computing $\mathbb{E}'$ we use $\sum_{j \neq i} Y_j$; in the right hand side of (3), we have the total variation distance between $\sum_{j \neq i} X_j$ and $\sum_{j \neq i} Y_j$—that is, the ℓ_1 distance between the distribution functions of these random variables.

Our first structural result for Nash equilibria in anonymous games comes from the following probabilistic lemma, bounding the right hand side of (3). To describe it, let us denote $[n] := \{1, \ldots, n\}$.

Theorem 4 ([12]) *Let $\{q_j\}_{j=1}^n$ be arbitrary probability values, and $\{X_j\}_{j=1}^n$ independent indicators with $\mathbb{E}[X_j] = q_j$, for all $j \in [n]$, and let z be a positive integer. Then there exists another set of probability values $\{\widehat{q}_j\}_{j=1}^n$ satisfying the following properties:*

1. *$||q_j - \widehat{q}_j|| = O(1/z)$, for all $j \in [n]$;*
2. *$\widehat{q}_j$ is an integer multiple of $\frac{1}{z}$, for all $j \in [n]$;*
3. *if $\{Y_j\}_{j=1}^n$ are independent indicators with $\mathbb{E}[Y_j] = \widehat{q}_j$, for all $j \in [n]$, then,*

$$\left\| \sum_{j} X_j - \sum_{j} Y_j \right\| = O(1/\sqrt{z}),$$

and, for all $i \in [n]$,

$$\left\| \sum_{j \neq i} X_j - \sum_{j \neq i} Y_j \right\| = O(1/\sqrt{z}).$$

Observe that the straightforward approach that rounds the q_i's into their closest multiple of $1/z$ is very wasteful, resulting in an upper bound of $n \cdot 1/z$ in the worst case. To get rid of this dependence on n from our bound, we resort instead to a delicate interleaving of the central limit theorem and the law of rare events—in fact, finitary versions thereof. The way these probabilistic tools become relevant in our setting is in approximating the aggregate behavior $\sum_j X_j$ of the players in an anonymous game with simpler distributions, Normal or Poisson depending on the X_j's.

Combining Theorem 4 with Bound (3), we get the following structural result.

Theorem 5 ([12]) *In every two-strategy anonymous game, there exists an ϵ-Nash equilibrium in which all players use strategies that are integer multiples of $O(\epsilon^2)$.*

It requires some thought, but we can turn Theorem 5 into an efficient algorithm for finding ϵ-Nash equilibria as follows: we perform a search over all *unordered* collections of n integer multiples of $O(\epsilon^2)$; for each such collection, we decide if there is an assignment of its elements to the players of the game that is an ϵ-Nash equilibrium—this latter task can be formulated as a max-flow problem and solved with linear programming. The overall running time of the algorithm is $n^{O(1/\epsilon^2)}$.

It turns out that the algorithm we just outlined can be a bit wasteful. To see this, suppose that in some Nash equilibrium all the players of the game use mixed strategies that are bounded away from 0 and 1. Then by the Berry-Esséen theorem [16], their aggregate behavior $\sum_j X_j$ should be close to a Normal distribution and, for this reason, close also to a Binomial distribution. Hence, in this case, we should be able to replace the X_j's by identical indicators and still satisfy the equilibrium conditions approximately by virtue of (3). On the other hand, if only a small number of players mix, we need to approximate their strategies very delicately, and Theorem 5 should become relevant in this case. The following theorem quantifies this intuition.

Theorem 6 ([14]) *There is a universal constant $c > 0$ such that, in every two-strategy anonymous game, there is an ϵ-Nash equilibrium in which, for $k = c/\epsilon$,*

1. *either at most k^3 players randomize, and their mixed strategies are integer multiples of $1/k^2$;*
2. *or all players who randomize use the same mixed strategy which is an integer multiple of $\frac{1}{kn}$.*

Observe that Case 2 in the statement of Theorem 6 is in direct correspondence to Nash's assertion for exact equilibria in symmetric games, guaranteeing the existence of a Nash equilibrium in which every player has the same mixed strategy (see Theorem 3). However, we now need to account for the possibility outlined in Case 1, that only a few players randomize; in this case, we resort to a use of Theorem 5.

With this refined characterization of ϵ-Nash equilibria in anonymous games given by Theorem 6, we obtain an algorithm with running time

$$\mathrm{poly}(n) \cdot (1/\epsilon)^{O(1/\epsilon^2)}.$$

Both this algorithm and the one based on Theorem 5 are of the *oblivious* kind, introduced in the context of approximation algorithms for 2-player games earlier in this survey. They both search over a fixed set of candidate solutions, and they only look at the input game to check if a candidate solution is an ϵ-Nash equilibrium. In fact, we need to make a slight modification to this definition in order to make it meaningful for (multi-player) anonymous games: we only require the set of candidate solutions to consist of unordered (i.e., not assigned to players) collections of mixed strategies. (Otherwise the set of candidate solutions would just explode!) Then, given a collection of mixed strategies, we only look at the game to decide if there is an assignment of its strategies to the players, corresponding to an ϵ-Nash equilibrium.

Does there exist a more efficient oblivious algorithm for anonymous games? In joint work with Papadimitriou [15], we show that the answer is essentially "no". And this establishes a remarkable parallel with 2-player games: the best approximation algorithm for both problems is oblivious, and its running time is optimal within the class of oblivious algorithms! So, are oblivious algorithms the only type of algorithms that are successful for equilibrium computation? While the answer remains unknown for 2-player games, the following result from [15] provides a non-oblivious algorithm for anonymous games, breaking the oblivious lower bound.

Theorem 7 ([15]) *There is a non-oblivious PTAS for two-strategy anonymous games with running time*

$$\mathrm{poly}(n) \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))}.$$

The new algorithm is based on the following idea: instead of searching the space of unordered collections of players' strategies, we could look at other aggregates of the players' behavior. To obtain Theorem 7, we search over collections of the first $O(\log(1/\epsilon))$ moments of the players' collective behavior; that is, we search over values of the vector $(\mathbb{E}[X], \mathbb{E}[X^2], \ldots, \mathbb{E}[X^{O(\log 1/\epsilon)}])$, where $X = \sum_j X_j$. But we need a probabilistic lemma assuring us that, once we have guessed the vector of moments correctly, we can recover an approximate equilibrium. The following theorem goes a long way towards showing this.

Theorem 8 ([15]) *Let $q_1, \ldots, q_n$ and $\widehat{q}_1, \ldots, \widehat{q}_n$ be two collections of probability values in $(0, 1/2]$. Let also $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ be two collections of independent indicators with $\mathbb{E}[X_j] = q_j$ and $\mathbb{E}[Y_j] = \widehat{q}_j$, for all $j \in [n]$. If for all $\ell = 1, \ldots, d$*

$$\mathbb{E}\left[\left(\sum_j X_j\right)^\ell\right] = \mathbb{E}\left[\left(\sum_j Y_j\right)^\ell\right],$$

then

$$\left\|\sum_j X_j \;-\; \sum_j Y_j\right\|_{\mathrm{TV}} \leq 20(d+1)^{1/4}2^{-(d+1)/2}. \tag{4}$$

Informally, Theorem 8 implies that the total variation distance between two sums of independent indicators scales inverse exponentially with the number of moments of the two sums that are equal. Hence, if we know the first $O(\log 1/\epsilon)$ moments of some Nash equilibrium $X_1, \ldots, X_n$, it is enough to take any other collection $Y_1, \ldots, Y_n$ of indicators with the same first moments to approximate the aggregate behavior of the players at the Nash equilibrium. This observation quickly leads to the algorithm described in Theorem 7.

With this algorithm, we conclude our exposition of approximation algorithms for multi-player anonymous games. We presented three probabilistic results; each of them gave new insight into the structure of ϵ-Nash equilibria, and a faster algorithm. The best algorithm (Theorem 7) runs in time polynomial in the number of players and quasi-polynomial in $1/\epsilon$. Whether there is a faster algorithm, and what probabilistic tool will unravel this algorithm is to be determined. . .

5 Conclusions

We provided a survey of recent results on the complexity of the Nash equilibrium, and the connection of this problem to the computation of Brouwer fixed points and the class PPAD. We also looked at the problem of computing approximate equilibria, and turned to symmetries in games and the way these affect the complexity of Nash equilibria. We mostly focused on a class of games with collapsed player identities, called anonymous, and provided a set of structural results for their Nash equilibria, along with efficient approximation algorithms for these games.

Our study explored the existence of centralized algorithms for computing equilibria. Hence, our negative result—that computing a Nash equilibrium is PPAD-complete—implies that, unless PPAD = FP, there are also no efficient distributed protocols by which the players of a game could discover a Nash equilibrium. This is true even if the players know the payoff function of the other players, and are capable of observing each other's strategies. But, what if a player can only observe her own payoff, while not being exactly sure about how many players play the game, what their strategies are, and how these affect her payoff? Are there stronger lower bounds in this setting? This avenue is worth exploring.

On the other hand, in terms of computing equilibria, centralized algorithms only make the existence of distributed protocols *possible*. For the class of anonymous games that we studied in this survey, for which we gave efficient approximation algorithms, are there also efficient, natural distributed protocols converging to equilibria? And, how should these exploit the symmetries of these games? Finally, what other classes of succinct multi-player games are appealing for applications and have good computational features?

Acknowledgments

We thank Sébastien Roch for feedback that helped improve this survey.

References

[1] M. Blonski. The Women of Cairo: Equilibria in Large Anonymous Games. *Journal of Mathematical Economics*, 41(3):253–264, 2005.

[2] H. Bosse, J. Byrka and E. Markakis. New Algorithms for Approximate Nash Equilibria in Bimatrix Games. *WINE*, 2007.

[3] G. W. Brown and J. von Neumann. Solutions of Games by Differential Equations. In *H. W. Kuhn and A. W. Tucker (editors), Contributions to the Theory of Games*, 1:73–79. Princeton University Press, 1950.

[4] X. Chen and X. Deng. Settling the Complexity of Two-Player Nash Equilibrium. *FOCS*, 2006.

[5] X. Chen, X. Deng and S. Teng. Computing Nash Equilibria: Approximation and Smoothed Complexity. *FOCS*, 2006.

[6] V. Conitzer and T. Sandholm. New Complexity Results about Nash Equilibria. *Games and Economic Behavior*, 63(2):621–641, 2008.

[7] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[8] C. Daskalakis, P. W. Goldberg and C. H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. *STOC*, 2006. To appear in the *SIAM Journal on Computing*, special volume for STOC 2006.

[9] C. Daskalakis, A. Mehta, and C. H. Papadimitriou. A Note on Approximate Nash Equilibria. *WINE*, 2006.

[10] C. Daskalakis, A. Mehta, and C. H. Papadimitriou. Progress in Approximate Nash Equilibria. *EC*, 2007.

[11] C. Daskalakis and C. H. Papadimitriou. Three-Player Games Are Hard. *ECCC Report*, 2005.

[12] C. Daskalakis and C. H. Papadimitriou. Computing Equilibria in Anonymous Games. *FOCS*, 2007.

[13] C. Daskalakis and C. H. Papadimitriou. Discretized Multinomial Distributions and Nash Equilibria in Anonymous Games. *FOCS*, 2008.

[14] C. Daskalakis. An Efficient PTAS for Two-Strategy Anonymous Games. *WINE*, 2008.

[15] C. Daskalakis and C. H. Papadimitriou. On Oblivious PTAS's for Nash Equilibrium. *STOC*, 2009.

[16] R. Durrett. *Probability: Theory and Examples*. Second Edition, Duxbury Press, 1996.

[17] K. Etessami and M. Yannakakis. On the Complexity of Nash Equilibria and Other Fixed Points (Extended Abstract). *FOCS*, 2007.

[18] E. J. Friedman and S. Shenker. Learning and Implementation on the Internet. *University of Rutgers working paper,* 1997.

[19] D. Gale, H. W. Kuhn and A. W. Tucker. On Symmetric Games. In *H. W. Kuhn and A. W. Tucker (editors), Contributions to the Theory of Games*, 1: 81–87. Princeton University Press, 1950.

[20] C. B. Garcia, C. E. Lemke and H. Lüthi. Simplicial Approximation of an Equilibrium Point of Noncooperative N-Person Games. *Mathematical Programming*, 4:227–260, 1973.

[21] I. Gilboa and E. Zemel. Nash and Correlated Equilibria: Some Complexity Considerations. *Games and Economic Behavior*, 1(1):80–93, 1989.

[22] M. D. Hirsch, C. H. Papadimitriou and S. A. Vavasis. Exponential Lower Bounds for Finding Brouwer Fixed Points. *Journal of Complexity*, 5(4):379–416, 1989.

[23] E. Kalai. Partially Specified Large Games. *WINE*, 2005.

[24] R. Kannan and T. Theobald. Games of Fixed Rank: a Hierarchy of Bimatrix Games. *SODA*, 2007

[25] M. J. Kearns, M. L. Littman and S. P. Singh. Graphical Models for Game Theory. *UAI*, 2001.

[26] L. G. Khachiyan. A Polynomial Algorithm in Linear Programming. *Soviet Mathematics Doklady*, 20(1):191–194, 1979.

[27] B. Knaster, C. Kuratowski and S. Mazurkiewicz. Ein Beweis des Fixpunktsatzes für n-dimensionale Simplexe. *Fundamenta Mathematicae*, 14:132–137, 1929.

[28] E. Koutsoupias and C. H. Papadimitriou. Worst-case Equilibria. *STACS*, 1999.

[29] C. E. Lemke and J. T. Howson, Jr. Equilibrium Points of Bimatrix Games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423, 1964.

[30] R. J. Lipton, E. Markakis and A. Mehta. Playing Large Games Using Simple Strategies. *EC*, 2003.

[31] N. Megiddo and C. H. Papadimitriou. On Total Functions, Existence Theorems and Computational Complexity. *Theoretical Computer Science*, 81(2): 317–324, 1991.

[32] I. Milchtaich. Congestion Games with Player Specific Payoff Functions. *Games and Economic Behavior*, 13:111–124, 1996.

[33] J. Nash. Non-Cooperative Games. *Annals of Mathematics*, 54(2):286–295, 1951.

[34] J. von Neumann. Zur Theorie der Gesellshaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

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

[36] N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani (eds.). *Algorithmic Game Theory*. Cambridge University Press, 2007.

[37] N. Nisan and A. Ronen. Algorithmic Mechanism Design. *STOC*, 1999.

[38] C. H. Papadimitriou. Algorithms, Games, and the Internet. *STOC*, 2001.

[39] C. H. Papadimitriou. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

[40] C. H. Papadimitriou and T. Roughgarden. Computing equilibria in multiplayer games. *SODA*, 2005.

[41] J. Renegar. On the Computational Complexity and Geometry of the First-Order Theory of the Reals, Parts I–III. *Journal of Symbolic Computation*, 13(3):255–299, 301–327, 329–352, 1992.

[42] J. Rosenmüller. On a Generalization of the Lemke-Howson Algorithm to Noncooperative N-Person Games. *SIAM Journal on Applied Mathematics*, 21(1):73–79, 1971.

[43] T. Roughgarden. *Selfish Routing and the Price of Anarchy*. MIT Press, 2005.

[44] R. Savani and B. von Stengel. Exponentially Many Steps for Finding a Nash Equilibrium in a Bimatrix Game. *FOCS*, 2004.

[45] H. E. Scarf. The Approximation of Fixed Points of a Continuous Mapping. *SIAM Journal on Applied Mathematics*, 15(5):1328–1343, 1967.

[46] H. Tsaknakis and P. G. Spirakis. An Optimization Approach for Approximate Nash Equilibria. *WINE*, 2007.

[47] R. Wilson. Computing Equilibria of N-Person Games. *SIAM Journal on Applied Mathematics*, 21(1):80–87, 1971.

[48] Pakistan causes YouTube outage for two-thirds of world. *USATODAY Online*, February 26, 2008, `http://www.usatoday.com/tech/world/2008-02-25-pakistan-youtube-block_N.htm`.

[49] Pakistan blocks YouTube website, *BBC News Online*, February 24, 2008, `http://news.bbc.co.uk/2/hi/south_asia/7261727.stm`.