CHAPTER 2

The Complexity of Finding Nash Equilibria

Christos H. Papadimitriou

Abstract

Computing a NASH equilibrium, given a game in normal form, is a fundamental problem for Algorithmic Game Theory. The problem is essentially combinatorial, and in the case of two players it can be solved by a pivoting technique called the Lemke–Howson algorithm, which however is exponential in the worst case. We outline the recent proof that finding a NASH equilibrium is complete for the complexity class PPAD, even in the case of two players; this is evidence that the problem is intractable. We also introduce several variants of succinctly representable games, a genre important in terms of both applications and computational considerations, and discuss algorithms for correlated equilibria, a more relaxed equilibrium concept.

2.1 Introduction

NASH's theorem – stating that every finite game has a mixed NASH equilibrium (Nash, 1951) – is a very reassuring fact: Any game can, in principle, reach a quiescent state, one in which no player has an incentive to change his or her behavior. One question arises immediately: Can this state be reached in practice? Is there an *efficient algorithm* for finding the equilibrium that is guaranteed to exist? This is the question explored in this chapter.

But why should we be interested in the issue of computational complexity in connection to NASH equilibria? After all, a NASH equilibrium is above all a conceptual tool, a prediction about rational strategic behavior by agents in situations of conflict – a context that is completely devoid of computation.

We believe that this matter of computational complexity is one of central importance here, and indeed that the algorithmic point of view has much to contribute to the debate of economists about solution concepts. The reason is simple: If an equilibrium concept is not efficiently computable, much of its credibility as a prediction of the behavior of rational agents is lost – after all, there is no clear reason why a group of agents cannot be simulated by a machine. Efficient computability is an important modeling

perequisite for solution concepts. In the words of Kamal Jain, "If your laptop cannot find it, neither can the market."[1]

2.1.1 Best Responses and Supports

Let us thus define NASH to be the following computational problem: Given a game in strategic form, find a NASH equilibrium. Since NASH calls for the computation of a real-valued distribution for each player, it seems *primae facie* to be a quest in continuous mathematics. However, a little thought reveals that the task is *essentially combinatorial*.

Recall that a mixed strategy profile is a NASH equilibrium if the mixed strategy of each player is a *best response* to the mixed strategies of the rest; that is, it attains the maximum possibly utility among all possible mixed strategies of this player. The following observation is useful here (recall that the *support* of a mixed strategy is the set of all pure strategies that have nonzero probability in it).

Theorem 2.1 *A mixed strategy is a best response if and only if all pure strategies in its support are best responses.*

To see why, assume for the sake of contradiction that a best response mixed strategy contains in its support a pure strategy that is not itself a best response. Then the utility of the player would be improved by decreasing the probability of the worst such strategy (increasing proportionally the remaining nonzero probabilities to fill the gap); this contradicts the assumption that the mixed strategy was a best response. Conversely, if all strategies in all supports are best responses, then the strategy profile combination must be a NASH equilibrium.

This simple fact reveals the subtle nature of a mixed NASH equilibrium: Players combine pure best response strategies (instead of using, for the same utility, a single pure best response) in order to create for other players a range of best responses that will sustain the equilibrium!

Example 2.2 Consider the *symmetric game* with two players captured by the matrix

$$A = \begin{pmatrix} 0 & 3 & 0 \\ 0 & 0 & 3 \\ 2 & 2 & 2 \end{pmatrix}$$

A game with two players can be represented by two matrices (A, B) (hence the term *bimatrix game* often used to describe such games), where the rows of A are the strategies of Player 1 and the columns of A are the strategies of Player 2, while the entries are the utilities of Player 1; the opposite holds for matrix B. A bimatrix game is called *symmetric* if $B = A^T$; i.e., the two players have the same set of strategies, and their utilities remain the same if their roles are reversed.

In the above symmetric game, consider the equilibrium in which both players play the mixed strategy $(0, 1/3, 2/3)$. This is a *symmetric* NASH equilibrium,

[1] One may object to this aphorism on the basis that in markets agents work in *parallel*, and are therefore more powerful than ordinary algorithms; however, a little thought reveals that parallelism cannot be the cure for exponential worst case.

because both players play the same mixed strategy. (A variant of NASH's proof establishes that every symmetric game, with any number of players, has a symmetric equilibrium – it may also have nonsymmetric ones.) We can check whether it is indeed an equilibrium, by calculating the utility of each strategy, assuming the opponent plays $(0, 1/3, 2/3)$: The utilities are 1 for the first strategy, and 2 for the other two. Thus, every strategy in the support (i.e., either of strategies 2 and 3) is a best response, and the mixed strategy is indeed a NASH equilibrium. Note that, from Player 1's point of view, playing just strategy 2, or just strategy 3, or any mixture of the two, is equally beneficial to the equilibrium mixed strategy $(0, 1/3, 2/3)$. The only advantage of following the precise mix suggested by the equilibrium is that it motivates the other player to do the same.

Incidentally, in our discussion of NASH equilibria in this chapter, we shall often use the simpler two-player case to illustrate the ideas. Unfortunately, the main result of this section says that two-player games are not, in any significant sense, easier than the general problem.

It also follows from these considerations that finding a mixed NASH equilibrium means finding the right supports: Once one support for each player has been identified, the precise mixed strategies can be computed by solving a system of algebraic equations (in the case of two players, linear equations): For each player i we have a number of variables equal to the size of the support, call it k_i, one equation stating that these variables add to 1, and $k_i - 1$ others stating that the k_i expected utilities are equal. Solving this system of $\sum_i k_i$ equations in $\sum_i k_i$ unknowns yields k_i numbers for each player. If these numbers are real and nonnegative, and the utility expectation is maximized at the support, then we have discovered a mixed NASH equilibrium.

In fact, if in the two-player case the utilities are integers (as it makes sense to assume in the context of computation), then the probabilities in the mixed NASH equilibrium will necessarily be rational numbers, since they constitute the solution of a system of linear equations with integer coefficients. This is not true in general: NASH's original paper (1951) includes a beautiful example of a three-player poker game whose only NASH equilibrium involves irrational numbers.

The bottom line is that *finding a NASH equilibrium is a combinatorial problem:* It entails identifying an appropriate support for each player. Indeed, most algorithms proposed over the past half century for finding NASH equilibria are combinatorial in nature, and work by seeking supports. Unfortunately, none of them are known to be efficient – to always succeed after only a polynomial number of steps.

2.2 Is the NASH Equilibrium Problem NP-Complete?

Computer scientists have developed over the years notions of complexity, chief among them *NP-completeness* (Garey and Johnson, 1979), to characterize computational problems which, just like NASH and SATISFIABILITY,[2] seem to resist efficient solution. Should we then try to apply this theory and prove that NASH is NP-complete?

[2] Recall that SATISFIABILITY is the problem that asks, given a Boolean formula in conjunctive normal form, to find a satisfying truth assignment.

It turns out that NASH is a very different kind of intractable problem, one for which NP-completeness is not an appropriate concept of complexity. The basic reason is that *every game* is guaranteed to have a NASH equilibrium. In contrast, in a typical NP-complete problem such as SATISFIABILITY, the sought solution may or may not exist. NP-complete problems owe much of their difficulty, and their susceptibility to NP-completeness reductions, to precisely this dichotomy.[3] For, suppose that NASH is NP-complete, and there is a reduction from SATISFIABILITY to NASH. This would entail an efficiently computable function f mapping Boolean formulae to games, and such that, for every formula ϕ, ϕ is satisfiable if and only if any NASH equilibrium of $f(\phi)$ satisfies some easy-to-check property Π. But now, given any unsatisfiable formula ϕ, we could guess a NASH equilibrium of $f(\phi)$, and check that it does not satisfy Π: This implies NP = coNP!

Problems such as NASH for which a solution is guaranteed to exist require much more specialized and subtle complexity analysis – and the end diagnosis is necessarily less severe than NP-completeness (see Beame et al., 1998; Johnson et al., 1988; Papadimitriou, 1994 for more on this subject).

2.2.1 NASH *vs* Brouwer

In contemplating the complexity of NASH, a natural first reaction is to look into NASH's proof (1951) and see precisely how existence is established – with an eye towards making this existence proof "constructive." Unfortunately this does not get us very far, because NASH's proof relies on *Brouwer's fixpoint theorem,* stating that every continuous function f from the n-dimensional unit ball to itself has a fixpoint: a point x such that $f(x) = x$. NASH's proof is a clever reduction of the existence of a mixed equilibrium to the existence of such a fixpoint. Unfortunately, Brouwer's theorem is well-known for its nonconstructive nature, and finding a Brouwer fixpoint is known to be a hard problem (Hirsch et al., 1989; Papadimitriou, 1994) – again, in the specialized sense alluded to above, since a solution is guaranteed to exist here also.

Natural next question: Is there a reduction in the opposite direction, one establishing that NASH is precisely as hard as the known difficult problem of finding a Brouwer fixpoint? The answer is "yes," and this is in fact a useful alternative way of understanding the main result explained in this chapter.[4]

2.2.2 NP-Completeness of Generalizations

As we have discussed, what makes NP-completeness inappropriate for NASH is the fact that NASH equilibria always exist. If the computational problem NASH is twisted

[3] But how about the traveling salesman problem? Does it not always have a solution? It does, but this solution (the optimum tour) is hard to verify, and so the TSP is not an appropriate comparison here. To be brought into the realm of NP-completeness, optimization problems such as the TSP must be first transformed into decision problems of the form "given a TSP instance and a bound B, does a tour of length B or smaller exist?" This problem is much closer to SATISFIABILITY.

[4] This may seem puzzling, as it seems to suggest that Brouwer's theorem is also of a combinatorial nature. As we shall see, in a certain sense indeed it is.

in any one of several simple ways that deprive it from its existence guarantee, NP-completeness comes into play almost immediately.

Theorem 2.3 (Gilboa and Zemel, 1989) *The following are NP-complete problems, even for symmetric games: Given a two-player game in strategic form, does it have*

- *at least two NASH equilibria?*
- *a NASH equilibrium in which player 1 has utility at least a given amount?*
- *a NASH equilibrium in which the two players have total utility at least a given amount?*
- *a NASH equilibrium with support of size greater than a given number?*
- *a NASH equilibrium whose support contains strategy s?*
- *a NASH equilibrium whose support does not contain strategy s?*
- *etc., etc.*

A simple proof, due to (Conitzer and Sandholm, 2003), goes roughly as follows: Reduction from SATISFIABILITY. It is not hard to construct a symmetric game whose strategies are all literals (variables and their negations) and whose NASH equilibria are all truth assignments. In other words, if we choose, for each of the n variables, either the variable itself or its negation, and play it with probability $\frac{1}{n}$, then we get a symmetric NASH equilibrium, and all NASH equilibria of the game are of this sort. It is also easy to add to this game a new pure NASH equilibrium (d, d), with lower utility, where d (for "default") is a new strategy. Then you add new strategies, one for each clause, such that the strategy for clause C is attractive, when a particular truth assignment is played by the opponent, only if all three literals of C are contradicted by the truth assignment. Once a clause becomes attractive, it destroys the assignment equilibrium (via other strategies not detailed here) and makes it drift to (d, d). It is then easy to establish that the NASH equilibria of the resulting game are precisely (d, d) plus all satisfying truth assignments. All the results enumerated in the statement of the theorem, and more, follow very easily.

2.3 The Lemke–Howson Algorithm

We now sketch the Lemke–Howson algorithm, the best known among the combinatorial algorithms for finding a NASH equilibrium (this algorithm is explained in much more detail in the next chapter). It works in the case of two-player games, by exploiting the elegant combinatorial structure of supports. It constitutes an alternative proof of NASH's theorem, and brings out in a rather striking way the complexity issues involved in solving NASH. Its presentation is much simpler in the case of symmetric games. We therefore start by proving a basic complexity result for games: looking at symmetric games is no loss of generality.

2.3.1 Reduction to Symmetric Games

Define SYMMETRIC NASH to be the following problem: Given a symmetric game, find a symmetric NASH equilibrium. As noted above, NASH proved in his original paper that such equilibrium always exists. Here we establish the following fact, which was actually first pointed out before NASH's paper, in Gale et al., 1950 essentially with the same proof, for the case of two-player zero-sum games:

Theorem 2.4 *There is a polynomial reduction from* NASH *to* SYMMETRIC NASH.

Thus the symmetric case of NASH is as hard as the general one.

We shall describe the reduction for the two-player case, the proof for any fixed number of players being a straightforward generalization. Suppose that we are given a two-player game described by matrices A and B; without loss of generality, assume that all entries of these matrices are positive (adding the same number to all entries of A or B changes nothing). Consider now the symmetric game consisting of this matrix: $C = \begin{pmatrix} 0 & A \\ B^T & 0 \end{pmatrix}$ and let (x, y) be a symmetric equilibrium of this game (by x we denote the first m components of the vector, where m is the number of rows of A, and by y the rest). It is easy to see that, for (x, y) to be a best response to itself, y must be a best response to x, and x must be a best response to y. Hence, x and y constitute a NASH equilibrium of the original game, completing the proof.

Incidentally, it is not known how hard it is to find *any* NASH equilibrium in a symmetric game (it could be easier than NASH), or to find a *nonsymmetric* equilibrium in a symmetric game (it could be easier *or* harder than NASH).

2.3.2 Pivoting on Supports

So, let us concentrate on finding a NASH equilibrium in a symmetric two-player game with $n \times n$ utility matrix A, assumed with no loss of generality to have nonnegative entries and in addition no column that is totally zero. Consider the convex polytope P defined by the $2n$ inequalities $Az \leq 1, z \geq 0$ (it turns out that these inequalities are important in identifying mixed NASH equilibria, because, intuitively, when an inequality from $A_i x \leq 1$ is tight, the corresponding strategy is a best response). It is a nonempty, bounded polytope (since $z = 0$ is a solution, and all coefficients of A are nonnegative while no column is zero). Let us assume for simplicity that the polytope P is also *nondegenerate*, that is, every vertex lies on precisely n constraints (every linear program can be made nondegenerate by a slight perturbation of its coefficients, so this is little loss of generality). We say that a strategy i is *represented* at a vertex z if at that vertex either $z_i = 0$ or $A_i z = 1$ or both – that is, if at least one of the two inequalities of the polytope associated with strategy i is tight at z.

Suppose that at a vertex z all strategies are represented. This of course could happen if z is the all-zero vertex – but suppose it is not. Then for all strategies i with $z_i > 0$ it must be the case that $A_i z = 1$. Define now a vector x as follows:

$$x_i = \frac{z_i}{\sum_{i=1}^{n} z_i}.$$

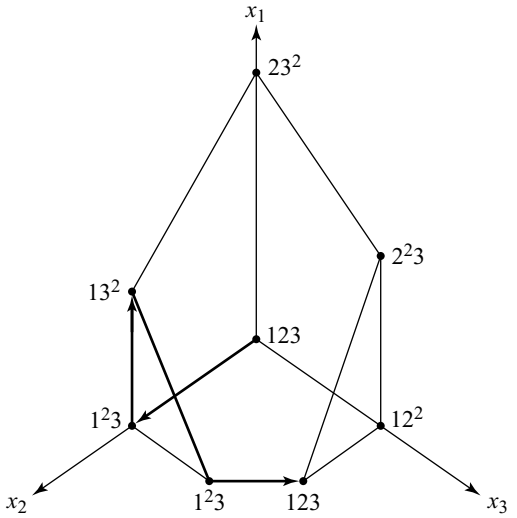

Figure 2.1. The Lemke–Howson algorithm can be thought of as following a directed path in a graph.

Since we assume $z \neq 0$, the x_i's are well defined, and they are nonnegative numbers adding to 1, thus constituting a mixed strategy. We claim that x is a symmetric NASH equilibrium. In proof, just notice that x satisfies the necessary and sufficient condition of a NASH equilibrium (recall Theorem): Every strategy in its support is a best response.

Let us apply this to the symmetric game of Example 2.2, with utility matrix

$$A = \begin{pmatrix} 0 & 3 & 0 \\ 0 & 0 & 3 \\ 2 & 2 & 2 \end{pmatrix}.$$

The polytope P is shown in Figure 2.1; it is nondegenerate because every vertex lies on three planes, and has three adjacent vertices. The vertices are labeled by the strategies that are represented there (ignore the exponents ^{2} for a moment). The only vertices where all strategies are represented are the vertex $z = (0, 0, 0)$ and the vertex $z = (0, 1/6, 1/3)$ – notice that the latter vertex corresponds to the NASH equilibrium $x = (0, 1/3, 2/3)$.

So, any vertex of P (other than $(0, 0, 0)$) at which all strategies are represented is a NASH equilibrium. But how do we know that such a vertex exists in general? After all, not all choices of n tight constraints result in vertices of a polytope. We shall develop a pivoting method for looking for such a vertex.

Fix a strategy, say strategy n, and consider the set V of all vertices of P at which all strategies are represented *except possibly for strategy* n. This set of vertices is nonempty, because it contains vertex $(0, 0, 0)$, so let us start there a path $\langle v_0 = 0, v_1, v_2, \ldots \rangle$ of vertices in the set V. Since we assume that P is nondegenerate, there are n vertices adjacent to every vertex, and each is obtainable by relaxing one of the tight inequalities at the vertex and making some other inequality tight. So consider the n vertices adjacent to $v_0 = (0, 0, 0)$. In one of these vertices, z_n is nonzero and all other variables are zero, so this new vertex is also in V; call it v_1.

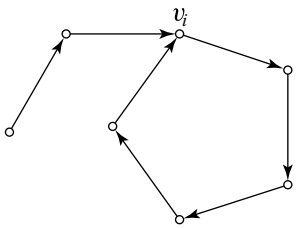

Figure 2.2. The path cannot cross itself.

At v_1 all strategies are represented except for strategy n, and in fact one strategy $i < n$ is "represented twice," in that we have both $z_i = 0$ and $C_i z = 1$. (We represent this by i^2). By relaxing either of these two inequalities we can obtain two new vertices in V adjacent to v_1. One of them is v_0, the vertex we came from, and the other is bound to be some new vertex $v_2 \in V$.

If at v_2 all strategies are represented, then it is a NASH equilibrium and we are done. Otherwise, there is a strategy j that is represented twice at v_2, and there are two vertices in V that are adjacent to v_2 and correspond to these two inequalities. One of these two vertices is v_1 and the other is our new vertex v_3, and so on. The path for the example of Figure 2.1 where strategy $n = 3$ is the one that may not be represented, is shown as a sequence of bold arrows.

How can this path end? No vertex v_i can be repeated, because repeating v_i (see Figure 2.2) would mean that there are *three* vertices adjacent to v_i that are obtainable by relaxing a constraint associated with its doubly represented strategy, and this is impossible (it is also easy to see that it cannot return to 0). And it cannot go on forever, since P is a finite polytope. The only place where the process can stop is at a vertex in V, other than 0 (a moment's thought tells us it has to be different from 0) that has no doubly represented strategy – that is to say, *at a symmetric NASH equilibrium!*

This completes our description of the Lemke–Howson algorithm, as well as our proof of NASH's theorem for two-player, nondegenerate games.

2.4 The Class PPAD

Let us dissect the existence proof in the previous section. It works by creating a graph. The set of vertices of this graph, V, is a finite set of combinatorial objects (vertices of P, or sets of inequalities, where all strategies are represented, with the possible exception of strategy n). This graph has a very simple "path-like" structure: All vertices have either one or two edges incident upon them – because every vertex $v \in V$ has either one or two adjacent vertices (depending on whether or not strategy n is represented in v). The overall graph may be richer than a path – it will be, in general, a set of paths and cycles (see Figure 2.3). The important point is that there is definitely at least one known endpoint of a path: the all-zero vertex. We must conclude that there is another endpoint, and this endpoint is necessarily a NASH equilibrium of the game.

We must now mention a subtle point: *the paths are directed*. Looking at a vertex in V, we can assign a direction to its incident edge(s), at most one coming in and at most

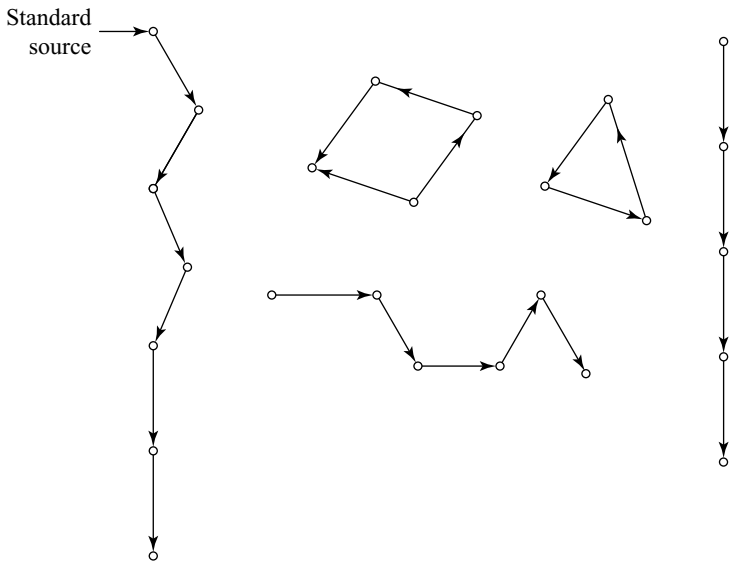

Figure 2.3. A typical problem in PPAD.

one going out, and do this in a way that is consistent from one vertex to another. In our three-dimensional example of Figure 2.1 the rule for asigning directions is simple: Going in the direction of the arrow, we should have a face all vertices of which are labeled 3 on our right, and a face all vertices of which are labeled 1 on our left. In games with more strategies, and thus a polytope of a higher dimension, there is a similar but more complicated (and more algebraic) "orientation rule." So, the graph in the proof of NASH's Theorem is a directed graph with all outdegrees and indegrees at most one.

What we mean to say here is that the existence proof of NASH's theorem (for the two-player symmetric, nondegenerate case, even though something similar holds for the general case as well) has the following abstract structure: A directed graph is defined on a set of nodes that are easily recognizable combinatorial objects (in our case, vertices of the polytope where all strategies, with the possible exception of strategy n, are represented). Each one of these vertices has indegree and outdegree at most one; therefore, the graph is a set of paths and cycles (see Figure 2.3). By necessity there is one vertex with no incoming edges and one outgoing edge, called a *standard source* (in the case of two-player NASH, the all-zero vertex). We must conclude that there must be a sink: a NASH equilibrium. In fact, not just a sink: notice that a source other than the standard (all-zero) one is also a NASH equilibrium, since all strategies are represented there as well. Another important point is that there is an efficient way, given a vertex in the graph to find its two adjacent vertices (or decide that there is only one). This can be done by simplex pivoting on the doubly represented variable (or on variable n, if it is represented).

Any such proof suggests a simple algorithm for finding a solution: start from the standard source, and follow the path until you find a sink (in the case of two-player NASH this is called the Lemke–Howson algorithm). Unfortunately, this is not an efficient algorithm because *the number of vertices in the graph is exponentially large*. Actually, in the case of two-player NASH there are examples of games in which such paths are exponentially long (Savani and von Stengel, 2004).

It turns out that, besides NASH, there is a host of other computational problems with guaranteed existence of solutions, for which existence follows from precisely this type of argument:

- A directed graph is defined on a finite but exponentially large set of vertices.
- Each vertex has indegree and outdegree at most one.
- Given a string, it is a computationally easy problem to (a) tell if it is indeed a vertex of the graph, and if so to (b) find its neighbors (one or two of them), and to (c) tell which one is the predecessor and/or which one is the successor (i.e., identify the direction of each edge).
- There is one known source (vertex with no incoming edges) called the "standard source."
- Any sink of the graph (a vertex with no outgoing edges), or any source other than the standard one, is a solution of the problem.

One problem whose existence proof has this form is finding an approximate Brouwer fixpoint of a function. We omit the precise definition and representation details here; a stylized version of this problem is defined in Section 2.6. Another is the following problem called HAM SANDWITCH: Given n sets of $2n$ points each in n dimensions, find a hyperplane which, for each of the n sets, leaves n points on each side. There are many other such problems (see Papadimitriou, 1994). For none of these problems do we know a polynomial algorithm for finding a solution.

All these problems comprise the complexity class called PPAD.[5] In other words, PPAD is the class of all problems, whose solution space can be set up as the set of all sinks and all nonstandard sources in a directed graph with the properties displayed above.

Solving a problem in PPAD is to telescope the long path and arrive at a sink (or a nonstandard source), fast and without rote traversal – just as solving a problem in NP means narrowing down to a solution among the exponentially many candidates without exhaustive search. We do not know whether either of these feats is possible in general. But we do know that achieving the latter would imply managing the former too. That is, P = NP implies PPAD = P (proof: PPAD is essentially a subset of NP, since a solution, such as a NASH equilibrium, can be certified quickly if found).

In the case of NP, we have a useful notion of difficulty – NP-completeness – that helps characterize the complexity of difficult problems in NP, even in the absence of a proof that P $\neq$ NP. A similar manoeuvre is possible and useful in the case of PPAD as well. We can advance our understanding of the complexity of a problem such as NASH by proving it *PPAD-complete* – meaning that all other problems in PPAD reduce to it. Such a result implies that we could solve the particular problem efficiently if and only if *all* problems in PPAD (many of which, like BROUWER, are well-known hard nuts that have resisted decades of efforts at an efficient solution) can be thus solved.

Indeed, the main result explained in the balance of this chapter is a proof that NASH is PPAD-complete.

[5] The name, introduced in Papadimitriou (1994), stands for "polynomial parity argument (directed case)." See that paper, as well as Beame et al. (1998) and Daskalakis et al. (2006), for a more formal definition.

2.4.1 Are PPAD-Complete Problems Hard?

But why do we think that PPAD-complete problems are indeed hard? PPAD-completeness is weaker evidence of intractability than NP-completeness: it could very well be that PPAD = P ≠ NP. Yet it is a rather compelling argument for intractability. If a PPAD-complete problem could be solved in polynomial time, then all problems in PPAD (finding Brouwer and Borsuk-Ulam fixpoints, cutting ham sandwiches, finding Arrow-Debreu equilibria in markets, etc., many of which have resisted decades of scrutiny, see Papadimitriou (1994) for a more complete list) would also be solved. It would mean that any local combinatorial description of a deterministic simplex pivoting rule would lead to a novel polynomial algorithm for linear programming. Besides, since it is known (Hirsch et al., 1989) that any algorithm for finding Brouwer fixpoints that treats the function as a black box must be exponential, PPAD = P would mean that there is a way to find Brouwer fixpoints by delving into the detailed properties of the function – a possibility that seems quite counterintuitive. Also, an efficient algorithm for a PPAD-complete problem would have to defeat the *oracles* constructed in Beame et al. (1998) – computational universes in which PPAD ≠ P – and so it would have to be extremely sophisticated in a very specific sense.

In mathematics we must accept as a possibility anything whose negation remains unproved. PPAD could very well be equal to P, despite the compelling evidence to the contrary outlined above. For all we know, it might even be the case that P = NP – in which case PPAD, lying "between" P and NP, would immediately be squeezed down to P as well. But it seems a reasonable working hypothesis that neither of these eventualities will actually hold, and that by proving a problem PPAD-complete we indeed establish it as an intractable problem.

2.5 Succinct Representations of Games

Computational problems have inputs, and the input to NASH is a description of the game for which we need to find an equilibrium. How long is such a description?

Describing a game in strategic form entails listing all utilities for all players and strategy combinations. In the case of two players, with m and n strategies respectively, this amounts to describing $2mn$ numbers. This makes the two-player case of NASH such a very neat and interesting computational problem.

But we are interested in games because we think that they can model the Internet, markets, auctions – and these have far more than two players. Suppose that we have a game with n players, and think of n as being in the hundreds or thousands – a rather modest range for the contexts and applications outlined above. Suppose for simplicity that they all have the same number of strategies, call it s – in any nontrivial game s will be at least two. *Representing the game now requires* ns^n *numbers!*

This is a huge input. No user can be expected to supply it, and no algorithm to handle it. Furthermore, the astronomical input trivializes complexity: If s is a small number such as 2 or 5, a trivial efficient algorithm exists: *try all combinations of supports*. But this algorithm is "efficient" only because the input is so huge: For fixed s, $(2^s)^n$ is polynomial in the length of the input, ns^n ...

Conclusion: In our study of the complexity of computational problems for games such as NASH we must be especially interested in games with many players; however, only *succinctly representable* multiplayer games can be of relevance and computational interest.

And there are many such games in the literature; we start by describing one of the latest arrivals (Kearns et al., 2001) that happens to play a central role in our story.

2.5.1 Graphical Games

Suppose that many players are engaged in a complex game; yet, the utility of each player depends on the actions of very few other players. That is, there is a directed graph $(\{1, 2, \ldots, n\}, E)$, with vertices the set of players, and $(i, j) \in E$ only if the utility of j depends on the strategy chosen by i (j's utility depends, of course, on the strategy chosen by j). More formally, for any two strategy profiles s and s' if $s_j = s'_j$, and, for all $(i, j) \in E$ we have $s_i = s'_i$, then $u_j(s) = u_j(s')$. A *graphical game,* as these are called, played on a graph with n nodes and indegree at most d, and s choices per player, requires only ns^{d+1} numbers for its description – a huge savings over ns^n when d is modest. (For more on graphical games, see Chapter 7.)

For a simple example, consider a directed cycle on 20 players, where the utilities are captured by the game matrix A of example 2.2. That is, if a player chooses a strategy $i \in \{1, 2, 3\}$ and his predecessor in the cycle chooses another strategy j, then the utility of the first player is C_{ij} (the utility of the predecessor will depend on the strategy played by *his* predecessor). Ordinarily, this game would require 20×3^{20} numbers to be described; its graph structure reduces this to just a few bytes.

Can you find a NASH equilibrium in this game?

2.5.2 Other Succinct Games

There are many other computationally meaningful ways of representing some interesting games succinctly. Here are some of the most important ones.

(i) *Sparse games*. If very few of the ns^n utilities are nonzero, then the input can be meaningfully small. Graphical games can be seen as a special case of sparse games, in which the sparsity pattern is captured by a graph whose vertices are the players.

(ii) *Symmetric games*. In a symmetric game the players are all identical. So, in evaluating the utility of a combination of strategies, what matters is *how many* of the n players play each of the s strategies. Thus, to describe such a game we need only $s\binom{n+s-1}{s-1}$ numbers.

(iii) *Anonymous games*. This is a generalization of symmetric games, in which each player is different, *but cannot distinguish between the others*, and so again his or her utility depends on the partition of the other players into strategies. $sn\binom{n+s-1}{s-1}$ numbers suffice here.

(iv) *Extensive form games*. These are given as explicit game trees (see the next chapter). A strategy for a player is a combination of strategies, one for each vertex in the game tree (information set, more accurately, see the next chapter for details) in which the player has the initiative. The utility of a strategy combination is that of the leaf reached if the strategies are followed.

(v) *Congestion games*. These games abstract the network congestion games studied in Chapters 18 and 19. Suppose that there are n players, and a set of *edges* E. The set of strategies for each player is a set of subsets of E, called *paths*. For each edge $e \in E$ we have a *congestion function* c_e mapping $\{0, 1, \ldots, n\}$ to the nonnegative integers. If the n players choose strategies/paths $P = (P_i, \ldots, P_n)$, let the *load* of edge e, $\ell(P)$ be the size of the set $\{i : e \in P_i\}$. Then the utility of the ith player is $\sum_{e \in P_i} c_e(\ell(P))$.

(vi) There is the even more succinct form of *network congestion games*, where E is the set of edges of an actual graph, and we are given two vertices for each player. The strategies available to a player are all simple paths between these two nodes.

(vii) *Local effect games*. These are generalizations of the congestion games, see Leyton-Brown and Tennenholtz 2003.

(viii) *Facility location games*. See Chapter 19.

(ix) *Multimatrix games*. Suppose that we have n players with m strategies each, and for each pair (i, j) of players an $m \times m$ utility matrix A^{ij}. The utility of player i for the strategy combination $s_1, \ldots, s_n)$ is $\sum_{j \neq i} A^{ij}_{s_i, s_j}$. That is, each player receives the total sum of his or her interactions with all other players.

2.6 The Reduction

In this section we give a brief sketch of the reduction, recently discovered in Daskalakis et al. (2006) and Goldberg and Papadimitriou (2006) and extended to two-player games in Chen and Deng (2005b), which establishes that NASH is PPAD-complete.

2.6.1 A PPAD-Complete Problem

The departure point of the reduction is BROUWER, a stylized discrete version of the Brouwer fixpoint problem. It is presented in terms of a function ϕ from the three-dimensional unit cube to itself. Imagine that the unit cube is subdivided into 2^{3n} equal cubelets, each of side $\epsilon = 2^{-n}$, and that the function need only be described at all cubelet centers. At a cubelet center x, $\phi(x)$ can take four values: $x + \delta_i$, $i = 0, \ldots, 3$, where the δ_is are the following tiny displacements mapping the center of the cubelet to the center of a nearby cubelet: $\delta_1 = (\epsilon, 0, 0)$ $\delta_2 = (0, \epsilon, 0)$, $\delta_3 = (0, 0, \epsilon)$, and finally $\delta_0 = (-\epsilon, -\epsilon, -\epsilon)$. If x is the center of a boundary cubelet, then we must make sure that $\phi(x)$ does not fall outside the cube – but this is easy to check. We are seeking a "fixpoint," which is defined here to be any internal cubelet corner point such that, among its eight adjacent cubelets, all four possible displacements δ_i, $i = 0, \ldots, 3$, are present.

But how is the function ϕ represented? We assume that ϕ is given in terms of a *Boolean circuit*, a directed acyclic graph of AND, OR, and NOT gates, with $3n$ bits as inputs (enough to describe the cublet in question) and two bits as outputs (enough to specify which one of the four displacements is to be applied). This is a computationally meaningful way of representing functions that is quite common in the complexity theory literature; any function ϕ of the sort described above (including the boundary checks) can be captured by such a circuit. And this completes the description of BROUWER, our starting PPAD-complete problem: Given a Boolean circuit describing ϕ, find a fixpoint

of ϕ. We omit the challenging proof that it is indeed PPAD-complete (see Daskalakis et al., 2006).

2.6.2 The Plan

But how should we go about reducing this problem to NASH? We shall start by reducing BROUWER to an intermediate graphical game with many players. All these players have just two strategies, 0 and 1; therefore, we can think of any mixed strategy of a player as a number in [0, 1] (the probability he or she assigns to strategy 1). Three of these players will be thought of as choosing three numbers that are the coordinates of a point in the cube. Others will respond by analyzing these coordinates to identify the cubelet wherein this point lies, and by computing (by a simulation of the circuit) the displacements δ_i at the cubelet and adjacent cubelets. The resulting choices by the players will incentivize the three original players to change their mixed strategy – *unless the point is a fixpoint of* ϕ, in which case the three players will not change strategies, and the graphical game will be at a NASH equilibrium!

2.6.3 The Gadgets

To carry out this plan, we need certain devices – commonly called "gadgets" in the reduction business – for performing basic arithmetic and logical operations. That is, we need to define certain small graphical games with players that are considered as inputs and another player as output, such that in any NASH equilibrium the mixed strategy of the output player (thought of as a real number between 0 and 1) stands in a particular arithmetical or logical relation with the inputs (again, thought of as numbers).

Consider, for example, the *multiplication game*. It has four players, two input players a and b, an output player c, and a middle player d. The underlying directed graph has edges (a, d), (b, d), (c, d), (d, c); i.e., one of these four players affects the utility of another if and only if there is an edge in this list from the former to the latter. The players have two strategies each, called 0 and 1, so that any mixed strategy profile for a player is in fact a real number in [0, 1] (the probability with which the player plays strategy 1). The utilities are so constructed that in any NASH equilibrium of this game, the output is always the product of the two inputs – all seen as numbers, of course: $c = a \cdot b$ (here we use a to represent not just player a, but also its value, i.e., the probability with which he plays strategy 1). To specify the game, we need to describe the utilities of the output and middle player (the utilities of the inputs are irrelevant since they have no incoming edges; this is crucial, because it allows the inputs to be "reused" in many gadgets, without one use influencing the others). If the middle player d plays 1 (recall that all nodes have two strategies, 1 and 0), then its utility is 1 if both inputs play 1, and it is 0 zero otherwise. Thus, if the two input players play 1 with probabilities a and b (recall that these are the "values" of the two inputs), and the middle player plays 1, then his utility is exactly $a \cdot b$. If on the other hand the middle player plays 0, then its utility is 1 if the output player plays 1, and it is 0 otherwise. Finally, the output player gets utility 1 if the middle player plays 1, and -1 if he plays 0.

Thus, the output player is motivated to play 1 with probability c, which is as high as possible, in order to maximize the utility from the middle player's playing 1 – but not

so high that the middle player is tempted to play 0, as he would whenever $c > a \cdot b$. Thus, at equilibrium, c must be exactly $a \cdot b$, and the multiplication gadget works!

In a similar manner we can construct gadgets that add and subtract their inputs (always within the range [0, 1], of course), or perform certain logical operations. For example, it is a trivial exercise to design a gadget with two nodes, an input x and an output y, such that $y = 1$ if $x > \frac{1}{2}$ and $y = 0$ if $x < \frac{1}{2}$ (notice that, importantly, the output of this comparator is undetermined is $x = \frac{1}{2}$). It is also easy to design gadgets that perform AND, OR, and NOT operations on their inputs (the inputs here are assumed to be *Boolean*, that is to say, pure strategies).

2.6.4 The Graphical Game

Using these devices, we can put together a graphical game whose NASH equilibria reflect accurately the Brouwer fixpoints of the given function ϕ.

The graphical game is huge, but has a simple structure: There are three players, called the *leaders*, whose mixed strategies identify a point (x, y, z) in the unit cube. These leaders are inputs to a series of comparators and subtractors which extract one by one the n most significant bits of the binary representation of x, y, and z, thus identifying the cubelet within which the point (x, y, z) lies. A system of logical gadgets could then compute the outputs of the given circuit that describes ϕ, when the inputs are the $3n$ extracted bits, repeat for the neighboring cubelets, and decide whether we are at a fixpoint.

But there is a catch: As we pointed out above, our comparators are "brittle" in that they are indeterminate when their input is exactly half. This is of necessity: It can be shown (see Daskalakis et al., 2006) that nonbrittle comparators (ones that behave deterministically at half) cannot exist! (It turns out that, with such comparators, we could construct a graphical game with no NASH equilibrium ...) This has the effect that the computation described above is imprecise (and, in fact, in an unpredictable manner) when the point (x, y, z) lies exactly on the boundary of a cubelet, and this can create spurious equilibria. We must somehow "smoothen" this discontinuity.

This is accomplished by a more complicated construction, in which the calculation of ϕ is carried out not for the single point (x, y, z) but for a large and very fine grid of points around it, with all results averaged.

Once the average displacement $(\Delta x, \Delta y, \Delta z)$ near (x, y, z) has been calculated, its components are added to the three leaders, completing the construction of the graphical game. This way the loop is closed, and the leaders (who had heretofore no incoming edges) are finally affected – very indirectly, of course – by their own choices. We must now prove that the NASH equilibria of this game correspond precisely to those points in the unit cube for which the average displacement is the zero vector. And from this, establish that the average displacement is zero if and only if we are near a fixpoint.

2.6.5 Simulating the Graphical Game by Few Players

We have already established an interesting result: Finding a NASH equilibrium in a graphical game is PPAD-complete. It is even more interesting because the underlying

directed graph of the game, despite its size and complexity, has a rather simple structure: It is *bipartite*, and all vertices have indegree *three* or less. It is bipartite because all gadgets are bipartite (the inputs and the outputs are on one side, the middle nodes on the other; the logical gadgets can be redesigned to have a middle node as well); and the way the gadgets are put together maintains the bipartite property. Finally, the middle nodes of the gadget are the ones of maximum indegree – three.

The challenge now is to simulate this graphical game by one with finitely many players. Already in Goldberg and Papadimitriou (2006) and Daskalakis et al. (2006), a simulation by four players was shown, establishing that NASH is PPAD-complete even in the four-player case. The idea in the simulation is this: Each of the four players "represents" many nodes of the graphical game. How players are represented is best understood in terms of a particular undirected graph associated with the graphical game, called the *conflict graph*. This graph is defined on the vertices of the graphical game, and has an edge between two nodes u and v if in the graphical game either (a) there is an edge between u and v, in either direction, or (b) there are edges from both u and v to the same node w. This is the *conflict graph* of the game; it should be intuitively clear that eventualities (a) and (b) make it difficult for the same player to represent both u and v, and so coloring the conflict graph and assigning its color classes to different players makes sense. The crucial observation is that *the conflict graph of the graphical game constructed in the reduction is four-colorable.*

So, we can assign to each of four players (think of them as "lawyers") all nodes (call them "clients") in a color class. A lawyer's strategy set if the union of the strategy sets of his clients, and so the clients can be represented fairly if the lawyer plays the average of their mixed strategies. Since the clients come from a color class of the conflict graph, the lawyer can represent them all with no conflict of interest (he or she should not represent two players that play against one another, or two players who both play against a third one). But there is a problem: A lawyer may neglect some clients with small payoffs and favor (in terms of weights in his mixed strategy) the more lucrative ones. This is taken care of by having the four lawyers play, on the side, a generalization of the "rock-paper-scissors game," at very high stakes. Since this game is known to force the players to distribute their probabilities evenly, all clients will now be represented fairly in the lawyer's mixed strategy; the four-player simulation is complete.

These results, up to the four player simulation, first appeared in the beginning of October 2005 (Goldberg and Papadimitriou, 2006; Daskalakis et al., 2006). It was conjectured in Daskalakis et al. (2006) that the 3-player case of NASH is also PPAD-complete, whereas the 2-player case is in P. Indeed, a few weeks later, two independent and very different simulations of the graphical game by *three* players appeared (Chen and Deng, 2005b; Daskalakis and Papadimitriou, 2005) thus proving the first part of this conjecture. The proof in Daskalakis and Papadimitriou (2005) was local, and worked by modifying the gadgets so that the conflict graph became three-colorable; this approach had therefore reached its limit, because for the graphical game to work the conflict graph must contain triangles. It was again conjectured in Daskalakis and Papadimitriou (2005) that the two-player case can be solved in polynomial time. In contrast, the proof in Chen and Deng (2005b) was more ad hoc and nonlocal, and was therefore in a sense more open-ended and promising.

A month later, a surprisingly simple two-player simulation was discovered (Chen and Deng, 2005a), thus establishing that even the two-player case of NASH is PPAD-complete! The intuitive idea behind this new construction is that many of the "conflicts of interest" captured in the conflict graph (in particular, the (b) case of its definition) happen to be unproblematic in this particular game: The two input nodes of a gadget cannot effectively "conspire" to improve their lot – and thus they could, in principle, be represented by the same (carefully programmed) lawyer. Thus, only two players are needed, corresponding to the two sides of the bipartite graphical game. The construction is now in fact a little more direct: there is no graph game, and the two players are constructed ab initio, with the gadgets, as well as the side game of rock–paper–scissors, built in.

2.6.6 Approximate Equilibria

Incidentally, this side game of rock–paper–scissors is the source of another difficulty that permeates all these proofs, and which we have not yet discussed: It only guarantees that the lawyers *approximately* balance the interests of their clients; as a result, the whole reduction, and the argument at each stage of the construction, must be carried out in terms of ϵ-*approximate* NASH *equilibria*. An ϵ-approximate NASH equilibrium is a mixed strategy profile such that no other strategy can improve the payoff *by more than an additive* ϵ. (Notice that an ϵ-approximate NASH equilibrium may or may not be near a true NASH equilibrium.) It is easy to see, in retrospect, that this use of approximation is inherently needed: Two-player games always have rational NASH equilibria, whereas games with more players may have only irrational ones. Any simulation of the latter by the former must involve some kind of approximation.

Now that we know that computing NASH equilibria is an intractable problem, computing approximate equilibria emerges as a very attractive compromise. But can it be done in polynomial time? The reduction described so far shows that it is PPAD-complete to compute ϵ-approximate NASH equilibria when ϵ is exponentially small (smaller than the side of the cubelet in the initial BROUWER problem, or 2^{-cn} for some $c > 0$, where n is the number of strategies). Starting from an n-dimensional version of BROUWER, the result can be strengthened up to an ϵ that is an inverse polynomial, (n^{-c}) (Chen et al., 2006).

There are some positive algorithmic results known for approximate NASH equilibria: $\frac{1}{2}$-approximate NASH equilibria are very easy to compute in two-player games (Daskalakis et al., in press) and an ϵ-approximate NASH equilibrium can be found in less than exponential time (more specifically, in time $n^{\frac{\log n}{\epsilon^2}}$) in arbitrary games (see Lipton et al., 2003). Discovering polynomial algorithms for computing ϵ-approximate NASH equilibria for ϵ between these values – possibly for arbitrarily small constant $\epsilon > 0$ – remains an important open problem.

2.7 Correlated Equilibria

Consider the symmetric game (often called *chicken*) with payoffs

$$\begin{pmatrix} 4 & 1 \\ 5 & 0 \end{pmatrix}$$

The payoffs are supposed to capture the situation in which two very macho drivers speed toward an intersection. Each has two options: Stop or go. There are two pure equilibria (me and you) and the symmetric mixed equilibrium $(1/2, 1/2)$. These three NASH equilibria create the following three probability distributions on the pure strategy profiles: $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{pmatrix}$

Consider however the following distribution: $\begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}$. It is not a NASH equilibrium; in fact, it is easy to see that there are no two mixed strategies for the two players that generate this distribution (in algebraic terms, the matrix is not of rank one). However, it *is* a rational outcome of the game, in the following more sophisticated sense: Suppose that a trusted third party draws from this distribution, and recommends to each player to play according to the outcome. (Coming back to the drivers story, this solution, randomizing between (stop, go) and (go, stop) is tantamount to a traffic signal.) If the lower left box is chosen, e.g., the recommendation is that Player 1 go and Player 2 stop (i.e., green light for Player 1). What is remarkable about this distribution of recommendations is that it is *self-enforcing*: If either player assumes that the other will follow the recommendation, his best bet is to actually follow the recommendation!

This motivates the following definition (Aumann, 1974): A *correlated equilibrium* is a probability distribution $\{p_s\}$ on the space of strategy profiles that obeys the following conditions: For each player i, and every two different strategies j, j' of i, conditioned on the event that a strategy profile with j as is strategy was drawn from the distribution, the expected utility of playing j is no smaller than that of playing j':

$$\sum_{s \in S_{-i}} (u_{sj} - u_{sj'}) p_{sj} \geq 0. \qquad (CE)$$

(Naturally, we also require that $p_s \geq 0$ and $\sum_s p_s = 1$.) Here by S_{-i} we denote the strategy profiles of all players except for i; if $s \in S_{-i}$, sj denotes the strategy profile in which player i plays j and the others play s. Notice that the inequalities express exactly the requirement that, if a strategy profile is drawn from the distribution $\{p_s\}$ and each player is told, privately, his or her own component of the outcome, and if furthermore all players assume that the others will follow the recommendation, then the recommendation is self-enforcing.

Notice also the following: If $p^i, i = 1, \ldots, n$, is a set of mixed strategies of the players, and we consider the distribution p_s induced by it ($p_s = \prod_i p^i_{s_i}$) then the inequalities (CE) state that *these mixed strategies constitute a mixed* NASH *equilibrium!* Indeed, for each i, j, j', equation (CE) states in this case that, if j is in i's support, then it is a best response. (If strategy j is not in the support, then the inequality becomes a tautology, $0 \geq 0$; if it is in the support, then we can divide by its probability the whole inequality, and the resulting inequality says that j is best response.) We conclude that any NASH equilibrium is a correlated equilibrium. In other words, the correlated equilibrium is a generalization of the NASH equilibrium, allowing the probabilities on the space of strategy profiles to be correlated arbitrarily. Conversely, NASH equilibrium is the special case of correlated equilibrium in which p_s's are restricted to come from a product (uncorrelated) distribution.

For example, in the drivers game, the (CE) inequalities are as follows:

$$(4-5)p_{11} + (1-0)p_{12} \geq 0$$
$$(5-4)p_{21} + (0-1)p_{22} \geq 0$$
$$(4-5)p_{11} + (1-0)p_{21} \geq 0$$
$$(5-4)p_{12} + (0-1)p_{22} \geq 0$$

A crucial observation now is that the (CE) inequalities are *linear* in the unknown variables $\{p_s\}$, and thus the system (CE) can always be solved efficiently by linear programming. In fact, we know that these inequalities always have at least one a solution: The NASH equilibrium that is guaranteed to exist by NASH's theorem.

To restate the situation in terms of our concerns in this chapter, *the correlated equilibrium is a computationally benign generalization of the intractable NASH equilibrium.* We can find in polynomial time a correlated equilibrium for any game. In fact, we can find the correlated equilibrium that optimizes any linear function of the $\{p_s\}$'s, such as the expected sum of utilities. For example, in the drivers game, we can optimize the sum of the players' expected utilities by maximizing the linear objective $8p_{11} + 6p_{12} + 6p_{21}$ over the polytope defined by the inequalities above. The optimum correlated equilibrium is this: $\begin{pmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 \end{pmatrix}$ – a traffic light that is red for both one third of the time.

2.7.1 Correlated Equilibria *vs* NASH Equilibria: The Whole Picture

The polytope defined by the (CE) inequalities in the case of the drivers game is shown in Figure 2.4 (the fourth dimension, $p_{22} = 1 - p_{11} - p_{12} - p_{21}$, is suppressed in the geometric depiction). Every point in this polytope is a correlated equilibrium. There are two pure NASH equilibria (N1 and N2) and one symmetric mied one (N3). The "traffic light" correlated equilibrium C1 $= \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}$ and the optimum one C2 $= \begin{pmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 \end{pmatrix}$ are also shown. *Notice that the three NASH equilibria are vertices of the polytope.* This is no coincidence.

Theorem 2.5 *In any nondegenerate two-player game, the NASH equilibria are vertices of the (CE) polytope.*

Naturally, not all vertices of the (CE) polytope will be NASH equilibria, but at least one will be. In other words, in two-player games every NASH equilibrium is the optimum correlated equilibrium for some linear function – unfortunately, guessing this function is apparently not easy.

To recapitulate, NASH equilibria are correlated equilibria satisfying the further constraint that they are the product distribution of some pair of mixed strategies. It is this single additional constraint that makes the problem of finding a NASH equilibrium so much harder. It is apparently a very nonconvex constraint (think of it as a curved surface in Figure 2.4, "touching" the (CE) polytope at three of its vertices). In contrast, for three or more players there are games in which the NASH equilibria are not vertices

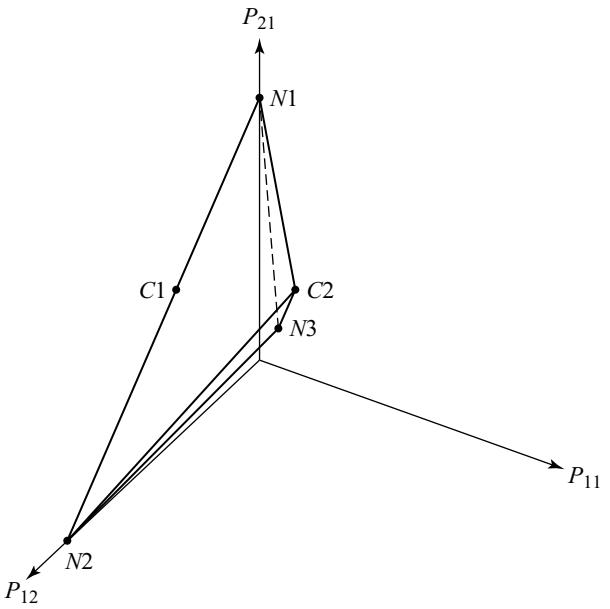

Figure 2.4. The three NASH equilibria (N1, N2, N3) of the drivers' game are vertices of the polytope of the correlated equilibria. Two other correlated equilibra are shown (C1, C2).

of the (CE) polytope; e.g., it is easy to see that any game with integer utilities that has only irrational NASH equilibria must be of this sort.

2.7.2 Correlated Equilibria in Succinct Games

But as we observed in Section 2.5, polynomial-time algorithms whose input is a game, such as the linear programming algorithm for finding correlated equilibria, make a mockery of complexity theory when the number of players is reasonably high. This brings us to the following important question: Can we find correlated equilibria efficiently when the game is represented succinctly?

There are some very interesting – and very natural – "learning" algorithms for *approximating* correlated equilibria, reviewed in Chapter 4 of this book. These algorithms work by simulating repeated play of the game, in which the various players change their strategies according to how much they "regret" previous decisions. Certain sophisticated ways of doing this are guaranteed to reach a point that is quite close to the (CE) polytope. To arrive at a distance ϵ, from the (CE) polytope, $\frac{1}{\epsilon^c}$ iterations are required, where c is some small constant depending on the particular method. But the question remains, can we find a point of the (CE) polytope in polynomial time?

Recently, there have been some interesting results on this question; to state them we need to introduce some definitions. We say that a succinctly representable game is *of polynomial type* if the number of players, as well as the number of strategies of each player, in a game represented by a string of length n is always bounded by a polynomial in n. For such a game, the *expected utility problem* is this: Calculate the expected utility of each player, if for each player i the given mixed strategy p^i played. It turns out

that solving this problem is enough for the correlated equilibrium problem to be solved:

Theorem 2.6 (Papadimitriou, 2005) *In any succinctly representable game of polynomial type for which the expected utility problem can be solved in polynomial time, the problem of finding a correlated equilibrium can be solved in polynomial time as well. Consequently, there is a polynomial-time algorithm (polynomial in the length of the description of the game) for finding a correlated equilibrium in sparse, symmetric, anonymous, graphical, congestion, local effect, facility location, and multimatrix games (among many others, recall the definitions in Section 2.5).*

But how about the slightly more demanding problem of finding, not just any correlated equilibrium, but the one that optimizes a given linear objective of the probabilities? A much less sweeping result is available here.

Theorem 2.7 (Papadimitriou and Roughgarden, 2005) *The problem of optimizing a linear function over correlated equilibria can be solved in polynomial time for symmetric games, anonymous games, and graphical games for which the underlying graph is of bounded treewidth.*

In contrast, it is NP-hard to find the optimum-correlated equilibrium in general graphical games and congestion games, among others (Papadimitriou and Roughgarden, 2005).

2.8 Concluding Remarks

The computational complexity of equilibrium concepts deserves a central place in game theoretic discourse. The proof, outlined in this chapter, that finding a mixed NASH equilibrium is PPAD-complete raises some interesting questions regarding the usefulness of the NASH equilibrium, and helps focus our interest in alternative notions (most interesting among them the approximate NASH equilibrium discussed in the end of Section 2.6).

But there are many counterarguments to the importance of such a negative complexity result. It only shows that it is hard to find a NASH equilibrium in some very far-fetched, artificial games that happen to encode Brouwer functions. Of what relevance can such a result be to economic practice?

The same can be said (and has been said, in the early days) about the NP-completeness of the traveling salesman problem, for example. And the answer remains the same: The PPAD-completeness of NASH suggests that any approach to finding NASH equibria that aspires to be efficient, as well as any proposal for using the concept in an applied setting, should explicitly take advantage of computationally beneficial special properties of the games in hand, by proving positive algorithmic results for interesting classes of games. On the other hand (as has often been the case with NP-completeness, and as it has started to happen here as well; Abbott et al., 2005; Codenotti

et al., 2006), PPAD-completeness proofs will be eventually refined to cover simpler and more realistic-looking classes of games. And then researchers will strive to identify even simpler classes.

An intractability result such as the one outlined in this chapter should be most usefully seen as the opening move in an interesting game.

Acknowledgment

Many thanks to Bernhard von Stengel for several useful suggestions.

Bibliography

T. Abbott, D. Kane, and P. Valiant. On the complexity of two-player win-lose games. *Proc. 2005 FOCS*.

R.J. Aumann. Subjectivity and correlation in randomized strategies. *J. Math. Econ.*, 1:67–96, 1974.

P. Beame, S. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57(1):13–19, 1998.

X. Chen and X. Deng. 3-NASH is PPAD-Complete. *Electronic Colloquium on Computational Complexity*, 134, 2005a.

X. Chen and X. Deng. Settling the complexity of 2-player Nash-equilibrium. *Electronic Colloquium on Computational Complexity*, 134, 2005b; *Fdns. Comp.* 2006, to appear.

X. Chen, X. Deng, and S. Teng. Computing Nash equilibria: Aprroximation and smoothed complexity. FOCS 2006, pp. 603–612, 2006.

B. Codenotti, M. Leoncini, and G. Resta. Efficient computation of Nash equilibria for very sparse win-lose games. *Electronic Colloquium on Computational Complexity*, 12, 2006.

V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In: *Proc. 18th Int. Joint Conf. Artificial Intelligence*, pp. 765–771, 2003.

C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The complexity of computing a Nash equilibrium. *Symp. on Theory of Computing*, 2006, pp. 71–78.

C. Daskalakis, A. Mehta, and C.H. Papadimitriou. A note on approximate Nash equilibria. In: *Proc. 2006 Workshop on Internet Network Economics*, in press.

C. Daskalakis and C.H. Papadimitriou. Three-player Games are Hard. *Electronic Colloquium on Computational Complexity*, 139, 2005.

F.S. Evangelista and T.E.S. Raghavan. A note on correlated equilibrium. *Intl. J. Game Theory*, 25(1):35–41, 2005.

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 Games,* 1:81–87. Princeton University Press, 1950.

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

I. Gilboa and E. Zemel. Nash and correlated equilibria: Some complexity considerations. *Games Econ. Behav.*, 1989.

P.W. Goldberg and C.H. Papadimitriou. Reducibility between equilibrium problems. *Symp. on Theory of Computing*, 2006, pp. 62–70.

S. Hart and D. Schmeidler. Existence of correlated equilibria. *Math. Operat. Res.,* 14(1):18–25, 1989.

M. Hirsch, C.H. Papadimitriou, and S. Vavasis. Exponential lower bounds for finding brouwer fixpoints. *J. Complexity*, 5:379–416, 1989.

D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988.

M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In: *Proc. Conf. on Uncertainty in Artificial Intelligence*, 2001, pp. 253–260.

K. Leyton-Brown and M. Tennenholtz. Local-effect games. *Intl. Joint Conf. Artificial Intelligence*, 2003, pp. 772–780.

R.J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. *ACM Electronic Commerce*, 2003, pp. 36–41.

J. Nash. Noncooperative games. *Ann. Math.*, 54:289–295, 1951.

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

C.H. Papadimitriou. Computing correlated equilibria in multi-player games. *Symp. on Theory of Computing*, 2005, pp. 49–56.

C.H. Papadimitriou and T. Roughgarden. Computing equilibria in multi-player games. *Symp. on Discrete Algorithms*, 2005, pp. 82–91.

R. Savani and B. von Stengel. Exponentially many steps for finding a Nash equilibrium in a Bimatrix Game. *Proc. of 45th Fdns. on Comp. Science*, pp. 258–267, 2004.

B. von Stengel. Computing equilibria for two-person games. *Handbook of Game Theory with Economic Applications*, Vol. 3, R. J. Aumann and S. Hart, eds. Elsevier, Amsterdam, pp. 1723–1759, 2002.