CENTRAL EUROPEAN POLITICAL SCIENCE REVIEW

2018
Winter

Quarterly of Central European Political Science Alliance

Volume: 19
Number: 74

CONTENT

GAME THEORIES

REPORTS AND CONFERENCES

BOOK REVIEW

INTRODUCTION

Our pleasure is to present to our readers the **Vol. 19. No. 74 of Central European Political Science Review (CEPSR)**, which main topic is: GAME THEORIES. Two studies are dealing with the game theory, both was presented at the John Harsányi Conference (in June, Budapest). (See the chapter of Janos Simon and Mohammad Farid Bin Abedin Bhuiyan).

The other studies are dealing with the cultural influence and political consequence of the past. The merit of the rule of the studies is the rule of the cultural dimensions in the present political life of Central European countries.

The current issue, **No. 74 of CEPSR** has authors from different countries, from USA, Bangladesh, Hungary, Italy, England.

One of the main goals of the journal editorial board of the **Central European Political Science Review** in to make it available to the broadest circle of readers from among experts and persons with a serious interest in the issues of the unique space of Central Europe, from the different perspective of history, international relations, political science, sociology, anthropology, art-sociology and data-analysis respectively.

One of the main reason for publishing the **CEPSR** is to serve and enhance Central Europeanism, and Europeanism in the science too.

We suggest you to visit our website: www.cepsr.eu or www.cepsr.hu and contact with the assistant our editor: kossuth.borbala@gmail.com

János Simon
Editor-in-chief

Janos Simon Chicago

Computational Complexity and Game Theory a short and incomplete tutorial

1 Introduction

Consider the relatively recent result[1]

Theorem 1 *Finding (approximate) Nash Equilibria (even in 2-Player Games) is PPAD-complete.*

I will start by a short reminder of Nash Equilibria, then will give a short intro- duction of what Computational Complexity aims to do, and explain the terms used in the statement above. Loosely speaking, they mean that there are games for which the task of actually finding their Nash Equilibria is computationally infeasible. I will define the terms precisely, and–I hope–in ways that are intuitive and clear. I will also try to give some idea of the flavor of the Mathematics and Computer Science involved, and will conclude with a brief discussion of the implications of the result. The aims of this research hark back to some of the concerns and results in related works of Hars´anyi's in Game Theory [9, 10, 11].

This survey can be read in many ways. I outline some possible roadmaps. The most impatient reader may just jump to the final Section 9 for a discussion of the possible implications of these results to Social Science, and deal with the rest of the paper at a later time. A more ambitious reader may still want to select which sections to read: (needless to say, I hope that the interested reader will find useful materials throughout the paper...) Section 2 is a review of basic concepts of Game Theory: subsection 2.1 presents the definitions of games, mixed strategies, and Nash Equilibria. The section ends with a restatement of the

fact that explicitly finding Nash Equilibria may be hard. This may be omitted or quickly skimmed by a reader familiar with these concepts.

Section 3 is an introduction to Computational Complexity, presenting the frame- work within which we can formally talk about computational hardness. It is a quick overview of the crucial definitions and intuitions. Introductory sections of textbooks on Algorithms, like [5, 12, 6] present this material in a formal and precise manner, at a more leisurely pace, and an interested reader may want to consult them. The overview is followed by a quick discussion of complexity classes in Section 4. In particular, we introduce several classes–*NP* and *PPAD*–that we believe are not computable within any reasonable timeframe. Unfortunately, we do not have *proofs* that these classes of problems are in fact infeasible. Thus, the claims of hardness in this paper are conditional: it is not impossible that these problems do, in fact have good algorithms we have not found yet–but this would be a revolutionary new discovery, a discovery that would go contrary to the expectation of most Theoreti- cal Computer Scientists. It would upend over at least half a century of research in Mathematics and Theoretical Computer Science. Again, it is possible to go on and read the rest of the paper without a careful reading of these sections, as long as one accepts that "PPAD-complete" means "incredibly hard".

Section 5 is, in a sense, tangential to the main argument. It illustrates the difficulty of proving hardness results in a much simpler context–and even there we are far from having a complete proof in every subcase that the problem is as hard as we believe it to be. The purpose of the section is to give the reader a feeling of why lower bound arguments are so difficult–in a sense, a Computer Scientist's excuse for not having real proofs of hardness. Again, the

main arguments of the paper can be grasped without reading this section.

The main arguments that show the difficulty of finding Nash Equilibria are pre- sented in the next few sections. In the brief Section 6 we define fixpoints, and outline the argument that Nash Equilibria are fixpoints. Nash's proof that Nash Equilibria always exist, is based on Brouwer's Fixpoint Theorem (Theorem 13) that guaran- tees the existence of fixpoints of certain functions. The next Section 7 introduces important tools used in the study of Nash Equilibria. In Subsection 7.1 we present the *End of the Line Problem* (see Definition 17) a combinatorial problem that is believed to be computationally infeasible. The rest of the paper, other than the concluding section, is devoted to the outline of the proof that the End of the Line problem is equivalent to the Nash Equilibrium Problem – in the sense that if we had a polynomial time algorithm for one, we could produce a polynomial time algorithm for the other. The precise statements are given in Theorem 18 in Subsection 7.2, and in Theorem 21 in Section 8. These theorems are proven through a series of reductions.

First, we introduce an important combinatorial theorem, Sperner's Lemma (The- orem 20.) Subsection 7.5 indicates how Sperner's Lemma is used to prove Brouwer's Fixpoint Theorem. Sperner's Lemma has a constructive proof, that uses essentially the End of the Line Problem. The outline of this constructive proof is given in Subsection 7.3. Finally, Subsection 7.6 brings all these constructions together, to show that one can find Nash Equilibria by solving the End of the Line Problem.

Section 8 offers a glimpse into the proof that one could solve the End of the Line Problem using a subroutine to find Nash Equilibria, completing the proof or our claim that Nash Equilibria and End of the Line are equally intractable.

The concluding Section 9 is a discussion of the implications for Social Science of the theorems discussed, and argues for informed interactions between computational experts and social scientists.

2 Game Theory

2.1 Games and Nash Equilibria

While I expect readers to be familiar with the definitions of games, strategies and mixed strategies, I repeat them briefly, for convenience. The reader who is familiar with these concepts may skip this section.

Deftnition 2 *A game for 2 players, whom we shall call* row player *and* column player *respectively, is specified by a list of* (pure) strategies $r_1, r_2, \cdots r_u$ *and* $c_1, c_2, \cdots c_v$ *for the row and column players, respectively, as well as a pair of* $u \times v$ payoff matrices P_{row} *for the row player, and and* P_{col} *for the column player.*

For every pair of pure strategies, r_i, s_j (the i-th strategy of the row player and the j-th strategy of the column player), $P_{row}(i, j)$ is the payoff that the row player receives if she uses her i-th strategy, and the column player uses his j-th strategy. $P_{column}(i, j)$ is the column player's payoff in the same situation.

Remarks.

1. The generalization to n players is straightforward, and well known. The infea- sibility results already hold for 2-player games, so there is no need for us to use more complicated games and notations in this survey.

2. In general P_{row} and P_{col} are unrelated. The special case when $P_{row} = -P_{col}$ is called a *zero sum* game.

Often, one presents the two matrices P_{row} and P_{col} as a single payoff matrix, where each entry is a pair of numbers (payoff of the row player, payoff of the column player).

A simple, well-known example is “rock, paper, scissors”, an old game of Chinese origin[2] where the two players simultaneously make hand gestures signifying the three objects. The rules are: paper wraps rock, scissors cut paper, and rock blunts scissors, yielding the payoff matrix below.

	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 Game Payoff Matrix

Deftnition 3 *A Nash Equilibrium is a pair of strategies such that it is not advan- tageous for either player to uni- laterally deviate from it.*

In other words, if *(rowstrategy , columnstrategy)* is a Nash Equilibrium, *rowstrat- egy* is the best response against *columnstrategy* (in the sense that any other strategy will yield a smaller payoff for the row player), and *columnstrategy* is the column player’s best strategy against *rowstrategy*.

It is easy to see that in the game above no pair of pure strategies is a Nash Equilibrium.

However, we allow *mixed strategies* where the row player can use a strategy $\sum_{i=1}^{u} p_i r_i$, with $\sum_{i=1}^{u} p_i = 1$ (that is, she will play strategy r_i, the i-th pure strategy, $\sum_{j=1}^{v} q_j s_j$, with $\sum_{j=1}^{v} q_j = 1$. Their (experted) playoffs will be $\sum_{i=1}^{u} \sum_{i=1}^{v} p_i q_j P_{row}(i,j)$ and $\sum_{i=1}^{u} \sum_{j=1}^{v} p_i q_j P_{column}(i,j)$, respectively. The reader can verify that the Rock-Paper-Scissors game above has a unique

Nash Equilibrium, where each player uses each of its strategies with probability 1/3, and has an expected payoff of 0.

Nash’s huge contribution to Game Theory was the theorem that bears his name, that states that using mixed strategies, equilibria always exist [15]. More precisely

Theorem 4 (Nash's Theorem) *Every game has at least one Nash Equilibrium.*

Remarks There is a potential difficulty, that, in fact we do not have to worry about: it is known that there exist 3-player Nash Equilibria where the probabilities are irrational numbers. This might look like a problem for computer solutions, which typically deal with numbers with a fixed (finite) number of digits. In the real world, it would be hard to think of situations where an approximation within, say .00000000000000000001, would be insufficient... In any case, one can define *s*- approximate Nash Equilibria, where deviation from the current strategy yield very small improvements. The theory (and the lower bounds of this survey) apply to such approximate equilibria. For the purposes of this paper, the reader may just think in terms of exact Nash Equilibria, although the proof outlined actually applies to (arbitrarily good) approximations.

2.2 Difficulties with Nash's Theorem

The existence of Nash Equilibria is a very powerful result. One may make a com- pelling argument that rational players should play strategies leading to a Nash Equilibrium. Still, several difficulties remain when one tries to use the result in applications:

1. Nash Equilibria may not be unique: a game may have many Nash Equilibria. It is not clear which of these we should chose (or which of these would be attained in a given situation or application modeled by a game.)

2. Nash's Theorem only guarantees the *existence* of a Nash Equilibrium. It does not provide us with an algorithm to find one.

Hars´anyi was well aware of these difficulties. In fact he also dealt with the situation – not discussed in this survey – "where players are uncertain about some important

parameters of the game situation" [9]. He has proposed some solutions based on Bayesian assumptions (and moral principles.) [9, 10, 11].

The results I want to discuss deal with the second difficulty in our list: actually finding Nash Equilibria. The results presented in this survey imply that finding Nash Equilibria may be computationally infeasible.

3 Computational Complexity

In Computer Science we do not look at a single computation, but at a family of problems. After all, the solution of a fixed instance of the problem can be precomputed, and we can just look it up. What we examine is the *rate of growth* of the resources (for the purpose of this paper, the relevant resource is *time*) needed, as the size of the instances grows. *Size* of the problem is the number *n*, of characters needed to specify it. Given some problem $\mathcal{Q}$, $TIME_{\mathcal{Q}}(n)$ is the biggest number of operations needed to solve any problem $\mathcal{Q}$ of size *n*. This is a *worst case* measure.
Computational Complexity Dogma A computation C is *feasible* if $TIME_C(n)$ is a polynomial.

Justifications:

- Reasonable models of computation are polynomially related (so doesn't mat- ter which computer we use: the times needed by one model will be at most polynomially bigger than the time needed by another)

- A program that uses a polynomial number of calls to a polynomial time pro- gram, runs in polynomial time

- If we increase our time budget by a factor of 10, we increase the size of the largest computation we can perform by some constant factor (for example, if we have a quadratic time algorithm, and we were able to solve problems of size up to N, with the improved time budget we will be able to solve problems of size $\sqrt{10} \times N$)

By contrast, an example of very, very nonfeasible time is the (non-polynomial) exponential function 2^n.

For this runtime, we have

- If we increase our time budget by a factor of 10, we increase the size of the largest computation by at most 4: if the previous biggest problem size was N , a factor 10 increase in time only lets us solve problems of size at most $N + 4$.

- Consider a computer that needs to perform 2^n operations, and is unbelievably fast: it can perform an operation in the time it takes light to go across the diameter of an electron. The computer would need *more than ten thousand times the age of the universe* to finish its computation on an input of length $n = 160$.[3]

4 P, NP, PPAD

We aggregate problems into large *complexity classes* according to the time needed to solve them. For simplicity, we will be looking at problems where the answer is YES or NO.[4]

Deftnition 5 *P – (deterministic) polynomial time computable – is the class of prob- lems that can be solved in polynomial time.*

This is the class of feasible problems we looked at in the previous section.

Deftnition 6 *NP – nondeterministic polynomial time – is the class of problems where we can verify the YES answers in polynomial time – assuming that we're given help. More precisely, there is a polynomial time algorithm A, a* verifier*, such that for any input x the following is true:*
(i) If the answer is YES, then there is a string h (a proof*) of length polynomial in x, such that A*(*x, h*) = 1. *(The verifier*

A attests that h is a valid proof that the answer to input x is YES.)

(i i) If the answer is NO, there is no string h such that A(*x, h*) = 1 *(The verifier does not accept purported proofs that the answer is YES, when the answer is NO).*

For example, consider the problem *Valid SUDOKU*: The input *x* is a 9 9 matrix, with some entries filled with numbers from 1 to 9. The answer is YES if this is a valid SUDOKU puzzle and NO if it isn't[5]. The problem is in the class *NP* : a correct solution to the puzzle is the required proof *h*–the verifier *A* simply checks that *h* is a valid solution that completes the initial values given in the matrix *x*. It is clear that such a "proof" exists if and only if the puzzle is valid.

A more mathematical example is *Graph G has a Hamiltonian cycle*. A Hamil- tonian cycle is a sequence of vertices connected by edges that comes back to the initial vertex, and visits every vertex exactly once. Again, this problem is in *NP* : the proof *h* is simply the Hamiltonian cycle. The verifier checks that the proof *h* is a valid sequence of vertices (consecutive vertices are linked by edges of the graph), no vertex repeats, and every vertex appears in the sequence.

In both examples, the "proof" *h* seems to have made the problem easier. In fact, we know of no polynomial time algorithm to determine whether a graph has a Hamiltonian cycle, while the task is trivial when *h* is given. The Hamiltonian Cycle Problem has a special status: *every* problem in *N P* can be solved in polynomial time, if it can use an algorithm that solves instances of the Hamiltonian Cycle Problem (and pay only 1 unit of time for each such use.) We say that every problem in *NP* can be *reduced* to the Hamiltonian Cycle Problem. Problems in *NP* with these special properties are called *N P*-complete. More formally

Deftnition 7 *A problem* $\mathcal{Q}$ *is* NP-complete *if*

- $\mathcal{Q}$ *is in NP, and*
- *Every problem in NP is reducible to* $\mathcal{Q}$

This means that is in $N\,P$, and *any* problem in $N\,P$ can be solved in (determin- istic) polynomial time given a subroutine that solves . Intuitively, the Hamiltonian Cycle Problem (or any $N\,P$-complete problem) is a *hardest* problem in $N\,P$: if only we could solve it in polynomial time, we could do anything in *NP* in polynomial time–so, in a sense, it contains the essence of what makes *NP* hard. There are many other *NP* -complete problems. It should be clear that all of them are likely to be very hard: if we could solve an *NP* -complete problem in polynomial time, we could solve the Hamiltonian Cycle Problem in polynomial time, which we do not believe is possible to do.

We believe that $P \subsetneq N\,P$, but we have no proof that $P \neq N\,P$. Proving this conjectured inequality is one of the big open problems of Mathematics and of The oretical Computer Science, and has been actively studied for over half a century. While it is widely believed that $P \neq N\,P$, it seems that a proof will require very substantial new techniques. In the next section I will try to give the interested reader a taste of why it is so very difficult proving such *lower bound* results, where one has to show the non-existence of efficient programs. Even apparently "obvious" and trivial lower bounds are hard to prove.

We are now ready to introduce the class PPAD used in the statement of the main theorem presented in this paper. First, we note that an alternative definition of the class *NP* is "the collection of the problems that are polynomial time reducible to the Hamiltonian Cycle Problem." We shall define the class *PPAD* through such an equivalence.

In Section 7.1 we define the *End of the Line Problem* (Definition 17). It is a problem in NP, not believed to be NP-complete, but not believed to be in P.

Deftnition 8 *PPAD – polynomial time parity argument on directed graphs – is the class of problems polynomial time reducible to the End of the Line Problem.*

Deftnition 9 *A problem E is* PPAD-compete *if it is in PPAD and every problem in PPAD in polynomial time reducible to it.*

Again, the this means that we believe that PPAD-complete problems cannot have polynomial time algorithms: we know that

$$P \subsetneq PPAD \subsetneq NP$$

We believe these are strict inclusions, that is, the true statement is that

$$P \subsetneq PPAD \subsetneq NP$$

If $P \subsetneq PPAD$, a $PPAD$-complete problem cannot have polynomial time algorithms. Thus, Theorem 1 of the Introduction means that finding Nash Equilibria is very unlikely to be in the class P. Stated informally: we believe that for some games computing Nash Equilibria is not feasible.

The reader is directed to Section 9 for a discussion of some of the implications.

The reader may wonder why we only "believe" these inequalities, instead of presenting a rigorous proof. Are Theoretical Computer Scientist incompetent? This may be the case, but the next section presents a kind of apology: these seem truly hard problems.

5 The Difficulty of Proving Lower Bounds

In this section I will try to illustrate why our ability to prove lower bounds is so unsatisfying.

A lower bound is an *impossibility result*–it states that some computational task cannot be performed with a certain amount of resources. Impossibility results are difficult to prove by their nature. How can we guarantee that something *cannot* happen? It is not an accident that in modern legal systems we do not have to prove that we *did not* commit a crime–it is the prosecutor's job to show that we *did*.

In the computational realm the difficulty is that we have to make sure that there are no unexpectedly clever, anti-intuitive algorithms, that behave much better than expected. To make these comments more precise, I will present a specific example of an apparently simple problem, with a seemingly obvious lower bound, where precise proofs end up being anything but simple.

5.1 Evasiveness

I assume that the reader is familiar with graphs. Recall that an (undirected) graph $tt = (V, E)$ has a finite set V of *vertices*, and a set E of *edges*. An edge is a pair of distinct vertices. We can ask about properties of such graphs, and about the complexity of deciding whether a given graph has a desired property.

For example, consider the property *graph G is connected* – meaning that from any vertex we can reach any other vertex by following a set of edges of the graph (the reader is invited to look for precise definitions in any book on Graph Theory, or Algorithms, for example [5]) So our computational problem is "Given a graph *tt*, what is the complexity of deciding whether it is connected?"

In order to talk about complexity, we must decide what do we mean by "Given a graph G", what model of computation, and what measure of complexity we wish to consider. First, consider a very simple mechanism, that in this

paper we will call the "advent calendar model" (computer scientists call it *decision tree complexity*). The algorithm can choose two of *tt*'s vertices, and ask an oracle whether there is an edge between them. The oracle will answer truthfully (this corresponds to finding a "window" in an advent calendar, and opening it, to see what message or picture it was hiding.) Now the algorithm can consider all it learned about the graph through this answer and all the previous answers, and decide what question to ask next. At some point, after asking a number of questions in this manner, the algorithm must declare, truthfully, whether *tt* is connected or not. The complexity of the algorithm is the number of questions asked. More precisely, for every natural number n, we define $f(n)$ to be the largest number of questions that the algorithm asks when the input is the graph with n vertices that forces the algorithm to ask the maximum number of questions.

Note that it suffices to ask $n(n \quad 1)/2$ questions: they will reveal all the edges of the graph, and so, having all the information about the graph, the algorithm will be able to make the correct decision. Must we ask this many questions?

Perhaps not surprisingly, the answer is yes. If an adversary oracle responds to queries, using the strategy "answer no, unless this makes it impossible for any graph compatible with the answers given so far to be connected" it will force the algorithm to ask all possible queries–it will be the answer to the last query that will determine whether the graph is connected or not. We say that the property of the graphs is *evasive* if we need (in the worst case) ask all queries. More precisely, for any algorithm that correctly decides whether graphs have the property, for every n there is an n-vertex graph tt_n for which the algorithm queries all edges[6].

This simple proof gives us the optimistic belief that we understand the complexity of the problem, at least within this simple model of computation, so, with some extra effort, we should be able to prove it in a general setting:

Conjecture 10 *Any nontrivial property of graphs is evasive.*[19]

By *nontrivial* we mean that there are graphs that have the property, and there are ones that don't. By *property of graphs* we mean that the property does not depend on the names we gave to the vertices – for example the property "vertex 1 has exactly one edge incident with it" is not a property of the graph–it is depends on which vertex we called 1. Technically, we require that the property be invariant under isomorphism of vertices (permuting the names of the vertices).

It turns out that the hypothesis is *false*.

Consider the *scorpion graph.*

It has a vertex of degree 1 (the sting), a vertex of degree 2 (the tail) joined to it by an edge. The tail is connected to a vertex of degree n 2 (the body), which is connected to the n 3 other vertices (the feet.) Some of the feet may have edges connecting pairs of feet. (Two vertices are connected if they have an edge between them.) See the figure below.

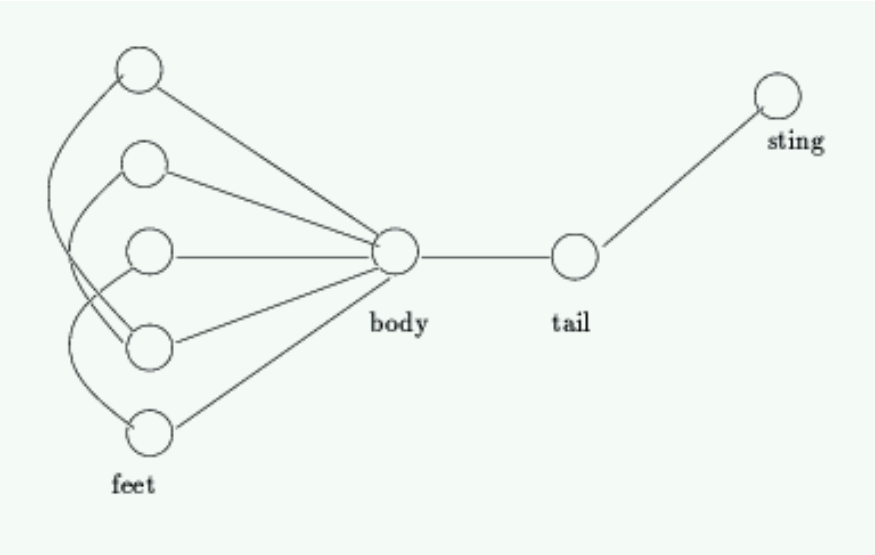

Figure 2: Scorpion Graph

There is an ingenious algorithm that asks only $O(n)$ queries to determine whether a graph is a scorpion graph. The reader is invited to find it![7]
So our Conjecture 10 is false. However, the following, revised conjecture is likely to actually be
true:

Conjecture 11 *Evasiveness Conjecture. Every monotone nontrivial graph property is evasive.*

A graph property is *monotone* if it does not become false if one adds edges to a graph that has the property.[8]
The difficulty of proving lower bounds – even in this case where input is so restricted, and complexity is defied in such straightforward manner is illustrated by the fact that the conjecture dates from 1973, and its current status is the following:
- It is known that at least $\Omega(n^2)$ (i.e. some constant times n^2 queries) are needed

- The conjecture has been proven for numbers that are prime, a power of a prime, and for some other classes of integers. Some of these proof use quite nontrivial Mathematics. [1]

In general, proving lower bounds for complicated problems is exceedingly difficult. Strategies of the form

"a natural way to solve the problem is to use strategy *S*, and we can prove that any algorithm based on this strategy must be inefficient"

only show that using strategy *S* is inefficient. Other strategies that rely on clever arguments we have may fare better–as exemplified by the algorithm that quickly recognizes scorpion graphs. Hardness proofs that rely on specific strategies are a proof of our ignorance (not being able to think of better strategies), not a proof of the difficulty of the problem.

After this interlude about why lower bounds are so challenging, let's get back to the main topic, showing that finding Nash Equilibria is hard.

6 Nash Equilibria as Fixpoints

Understanding the complexity of Nash Equilibria relies on a series of reductions–showing that finding a Nash Equilibrium reduces to a sequence of other combinatorial problems. First, we show that Nash Equilibria are special fixpoints.

Deftnition 12 *x is a* fixpoint *of function f* () *iff f* (*x*) = *x*

Obviously, the definition makes sense only if *f* () maps some domain into itself. We can view such functions as "moving" the point *x* to the point *f* (*x*), hence the definition of fixpoint – it is an input to the function that is not moved.

A function may have no fixpoints – for example consider $f(x) = x + 1$. On the other hand, "nice" (continuous) functions defined on suitable "nice" domains (compact and convex subsets of Euclidean spaces, like a disk on the plane) do. More precisely we have

Theorem 13 Brouwer's Theorem *Every continuous function from a compact and convex Euclidean space to itself has a fixpoint.*

Recall that *compact* means closed and bounded, where *closed* means that the limit of a sequence of elements of the set is contained in the set, and *convex* means that if two points are in the set, so is every point in the line connecting them.

Consider a 2-player game, where the row player has (pure) strategies $r_1, \cdots r_u$ and the column player has strategies $c_1 \cdots r_v$. Then a pair of strategies that specifies the (mixed) strategy of each player is defined by $x = p_1, p_2, \cdots p_u, q_1, q_2, \cdots q_v$ (the probabilities of each strategy for each of the two players). We can view x as a poi$\sum_{i=1}^{u} p_i = 1$ and $\sum_{j=1}^{v} q_j = 1$. u+v-dimensional space R^{u+v} where

Fix a small positive number s and suppose that there is some "preference function" $f(x$,x'hat chx'ges the strategies to a point where is a small change in the strategy of one of the players that improves her revenue. If x were a Nash Equilibrium, no such move would exist, and we would have $f(x) = x$. In other words, a Nash Equilibrium *is* a fixpoint of such a function. Nash essentially showed this function exists, it is continuous, and the subset B of the previous paragraph is compact and convex. He then used Brouwer's Theorem to conclude that a fixpoint must always exist.[9]

7 Computing Fixpoints with Sperner's Lemma and the End of the Line Problem

Nash's proof does not give us an efficient method to compute Nash Equilibria, the fixpoint we seek. It is a proof of existence, not an algorithm that returns the solution. Let use view some other examples of such inefficient proofs of existence.

7.1 The End of The Line Problem

Deftnition 14 *A directed graph* $G = (V, E)$ *is a set* V *of* vertices*, and a set of* directed edges E*. Each directed edge* $e \in E$ *is a pair* (u, v) *of vertices: e* goes from u to v*. The* outdegree *of vertex u,* $d_o(u)$*, is the number of edges of the form* $(u, v) \in E$ (*edges leaving* u)*, its* indegree $d_i(u)$*, is the number of edges of the form* (v, u) (*edges going into* u.)

Observation 15 *In any directed graph* $\sum_{v \in V} d_i(v) = \sum_{v \in V} d_o(v)$

This is true because each directed edge contributes 1 to each of the sums.

Corollary 16 *If a directed graph G has a vertex v such that* $d_o(v) > d_i(v)$ *then there must be another vertex u such that* $d_i(u) > d_o(u)$

In particular, if for all vertices w, $d_o(w) \leq 1$ and there is a vertex v with $d_i(v) = 0$ (a vertex of indegree 0) then there must be another vertex u with $d_o(u) = 0$ (a vertex of outdegree 0). This fact has a nice intuitive description: consider a graph and a vertex v with indegree 0. Consider the process where we go from each vertex a to the vertex b such that (a, b) is a directed edge, starting with vertex v. We must end with some vertex u where we cannot go further. In other words, in graphs where there is a vertex of indegree 0, we can trace out, from this vertex, a *line* – a directed path – that must end at some vertex (a vertex of outdegree 0). This vertex is the "end of the line."

Deftnition 17 *The* End of the Line Problem *is, given a directed graph* $G = (V, E)$*, and a vertex* $v \in V$ *of indegree* 0 *find a vertex u of outdegree* 0.

We believe that this problem is quite difficult. It is important to understand the details of the model: we assume that each vertex of the set V of vertices can be described by an n-digit number. There are 10^n such numbers from 0 to the number represented by n consecutive 9s, so V maybe exponentially big. Since it may also be the case that the line passes through every vertex of V before ending, the naive strategy of "following the line" can take exponential time.

Note that if the graph is explicitly given as input–say by a list of its edges–the problem is easy, as the length of the line to follow in the naive strategy is bounded by the length of the input. What makes the problem challenging is that the input graph is not directly available: what we have is the ability to decide for each vertex u whether it is a vertex of outdegree 0, or, if not, what is the name of the unique vertex v that the edge outgoing from u leads to.

Again, the difficulty of the problem is well understood if this information is given through the "advent calendar mechanism" – where all we can do is to choose a vertex w and make the query of where the outgoing edge leads to. Recall that the answer "to no other vertex" tells us that the outdegree os w is 0. It can be shown that in this model, the best algorithm to solve the problem is the naive strategy of following the path one vertex at a time, and this takes exponential time in the worst case.

The problem is that in our application (as we discussed, we will use the the End of the Line algorithm as a subroutine in our procedure to find a Nash Equilibrium) the answers to queries are computed by an explicit algorithm. The algorithm for vertex names of length n will be a Boolean circuit of size polynomial in n, that given u, the name of a vertex,

tells us the name of the vertex *v* that the outgoing edge from *u* leads to (or that no such vertex exists–*u* is the end of the line.) Now, instead of getting our information about the graph from an inscrutable black box, we have access to this circuit, and instead of blindly exploring the graph, a clever analysis of the circuit may reveal properties of the graph that could make our search more efficient.

The existence of such clever shortcuts may at first seem almost plausible, as only an exponen- tially small fraction of the possible graphs can be specified with polynomial size circuits. More precisely, the number of graphs with vertices described by *n*-digit numbers, with each vertex having outdegree and indegree bounded by 1 is doubly exponential, while the number of graphs that can be specified by a polynomial size circuit is exponential in a polynomial.[10]

While it is true that the graphs specified by small circuits are somehow special, we have not been able to take advantage of this property. Most computer scientists believe that it is extremely unlikely that we will be able to do so. It is notoriously hard to extract information from the specification of a computing device: for example we know that there cannot exist an algorithm that. given the text of a program and the input to the program, could tell whether the program with the given input will ever halt[11]. Our inability to discover even such fundamental properties of a program or of a circuit justifies our pessimism about finding clever shortcuts for the End of the Line problem, given the circuits that specify the graph. Consequently, there is widespread agreement that there is no efficient algorithm that solves the End of the Line Problem.

7.2 Nash Equilibria are no Harder than End of the Line

Nash Equilibria *would have* an efficient algorithm, were the End of the Line problem easy. The technical statement is

Theorem 18 *There is a polynomial time reduction of Nash Equilibria to End of the Line.*

In other words, there is an algorithm for Nash Equilibria that uses End of the Line as a subroutine, and if we count each use of this subroutine as a constant time operation, the algorithm runs in polynomial time.

Later, we will claim that, given an instance of the End of the Line problem, we can construct–in polynomial time–a game, such that finding a Nash Equilibrium of the game allows us to get the solution to the End of the Line problem. So, were Nash Equilibria easy to find, we would have an efficient algorithm for End of the Line.

The following subsections outline the techniques needed to prove Theorem 18. The proof is sketched in subsection 7.6.

7.3 From End of the Line to Fixpoints

As we saw, Nash's Theorem relied on the existence of a fixpoint guaranteed by Brouwer's Fixpoint Theorem. The proof of Brouwer's Theorem, in its turn, relies on a combinatorial theorem, Sperner's Lemma. I will present the statement and an outline of the proof of Sperner's Lemma, because the proof illustrates how solving the End of the Line problem is an essential part of the solution that finds a Nash Equilibrium.

Consider a triangle on the plane, that is triangulated–that is, it is subdivided into smaller triangles in a way that any two triangles meet only by sharing a vertex, and possibly by sharing an edge (in other words, no edge of any triangle has a vertex of another triangle, except possibly at its endpoints). Now associate one of three *colors* with each vertex according to the rule below:

- The vertices of the original outer big triangle are assigned the colors 1, 2, and 3[12]

- On each of the three lines connecting two of the vertices of this original outer big triangle, the vertices on it cannot receive the color of the third vertex of the outer triangle. In other words, on the line connecting vertices 1 and 2, no vertex can be colored 3, on the line connecting 1 and 3, no vertex can be colored 2 (all vertices on that line must have color 1 or color 3), and vertices on the outer line connecting vertex 2 and vertex 3 cannot be colored with color 1.

- All other vertices can be colored with any color.

We call a triangulation satisfying all the conditions above a *Sperner Triangulation*

Deftnition 19 *A triangle is* trichromatic *if it has vertices of all three colors.*

In other words, one of its vertices has the color 1, another the color 2. and one has the color 3.

Theorem 20 Sperner's Lemma *Every Sperner Triangulation has a (small) trichromatic triangle.*

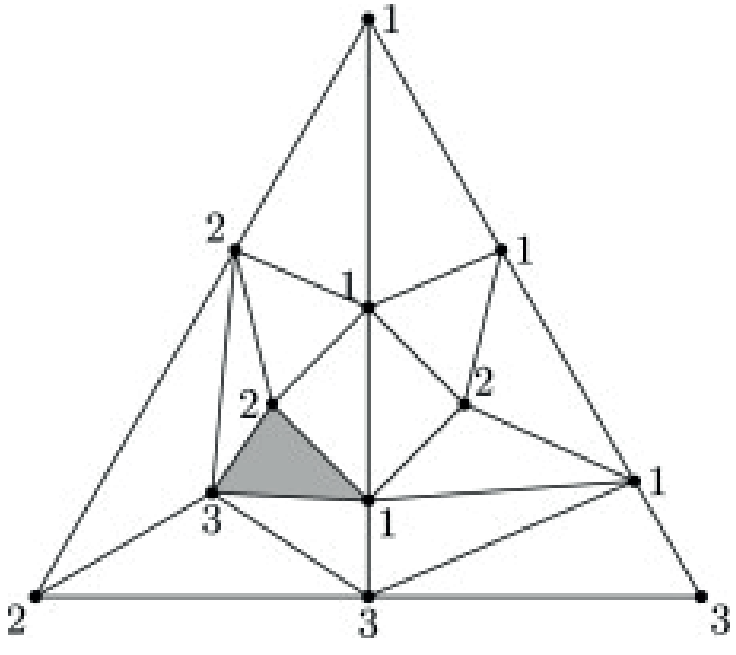

Figure 3: A Sperner triangulation and a trichromatic triangle

Figure 3 illustrates an instance of Sperner's Lemma. The shaded triangle is trichromatic. It is the only trichromatic triangle in this triangulation.[13] In general, a Sperner Triangulation may have many trichromatic triangles–the lemma guarantees that there is at least one.

Proof of Sperner's Lemma (sketch): Consider the line that connects vertices 1 and 2 of the original (big) triangle. Add an additional edge from each of the vertices in this line to the original vertex colored 2 (except for its neighbor, that already has such an edge.) Now build a new graph *H* as follows: put a vertex in the interior of each triangle, and a special vertex representing the outside area. If there is an edge of the triangulation with endpoints colored 1 and 2, put a directed edge between the two vertices in the adjacent triangles. The direction of the edge should be such that if we were to walk in the direction of the edge, the vertex colored 1 is to our left.

This corresponds to considering each edge of the triangulation connecting vertices colored 1 and 2 as a door, that we can pass through as indicated (keeping vertex labelled 1 to our left.)

The graph *H* has the following properties:

1. the vertex representing the outside area has indegree 0 and outdegree 1
2. no vertex has outdgree greater than 1, or indegree greater than 1

The second property follows because a triangle can have at most two edges with endpoints colored 1 and 2, and if this occurs, the corresponding doors will have opposite directions–one into the triangle and one out of it.

So the directed graph *H* must have a vertex with indegree 1 and outdegree 0 (as we have seen in our previous discussion

about graphs representing paths.) This vertex must represent a triangle of the subdivision – and *it must be a trichromatic triangle.* This is because in order to enter it, it must have an edge with endpoints labelled 1 and 2 – but if the third vertex was not labelled 3 the triangle would have two edges with endpoints 1 and 2, and, therefore, two doors– one into it and one out–and therefore have both indegree and outdegree 1. This cannot happen, as this triangle, by assumption, corresponds to a vertex with outdegree 0.

The next figure illustrates the construction.

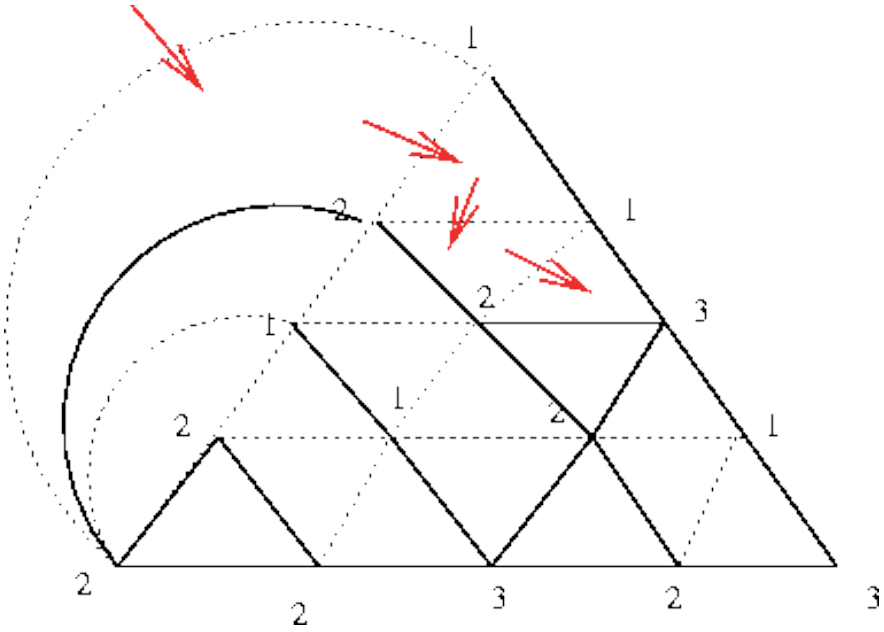

Figure 4: Going through the "gates" to a trichromatic triangle

Note that the proof above is *constructive*: it not only shows that a trichromatic triangle exists– it gives us a method to find it. The path in the graph *H* built in the proof leads us to the trichromatic triangle. However, note that the path can be as long as the number of triangles in the Sperner Triangulation.

7.4 Sperner's Lemma In Higher Dimensions

What we stated and proved was actually a special case of Sperner's Lemma, in dimension 2. Actually, the lemma is more general and is true in higher dimensional spaces. It is this more general result that we need to use for our proofs. However, the ideas are essentially the same, the proofs are similar, only the formalism is more cumbersome. The reader who just wants to have a "feel" for the proof is well served by the 2-dimensional "toy model" we presented.

In the general case one needs to substitute "triangles" by their high-dimensional equivalents – "regular polytopes" – for example in 3 dimensions the role of triangles would be performed by tetrahedra, and corresponding to triangulations we have subdivisions of a tetrahedron into smaller tetrahedra. As a tetrahedron has 4 vertices, the number of colors is 4, and the lemma asserts the existence of a small tetrahedron with all 4 colors at its vertices. The structure of the proof is identical to the 2-dimensional one: there are "doors" through the faces of adjacent tetrahedra, and a path leading to and ending at a 4-chromatic tetrahedron of the subdivision.

7.5 From Sperner's Lemma to Fixpoints

We want to show how to compute fixpoints using Sperner's Lemma. Consider a "nice" function[14] f that maps points of a triangle on the plane to itself. (Note to the reader: we are making several inessential simplifications here. We are looking at the case of 2-dimensional domains. This restriction of the dimension is, as stated earlier, only for the sake of a clearer exposition: in two dimensions we have more easily understandable geometric objects. We consider the domain to be a large triangle. Again, this is not a problem, as we can transform a closed bounded domain in the plane

to a triangle by a continuous transformation.) Our argument will actually only find an approximate fixpoint.[15]

Consider a triangulation of our domain into small triangles (say, no small triangles have a side of length exceeding some small parameter s). Each vertex of the triangulation is a point of the plane. At this point, w, consider the expression $f(w)\ w$. This is a vector on the plane, pointing to some direction, unless w is a fixpoint, when $f(w)\ w = 0$. If the fixpoint is in the interior of a triangle and the vertices of the triangle are close enough to it, one expects that the three vertices will have vectors pointing away from w in different directions. In particular, if we partition the directions in the plane into 3 classes – say the class 1, of angles from 0 to 120 degrees, the class 2, of angles from 120 to 240 degrees, and class 3, from 240 to 360 degrees – the three vertices of the triangle would belong to three distinct classes.

This is the connection to Sperner's Lemma. Consider the triangulation above, and color each vertex as we just discussed (compute the vector $f(v)\ v$ for each vertex v, and give the vertex v one of the colors 1, 2, or 3, according to the color class of the direction of the vector, as defined in the previous pragraph.) Now Sperner's Lemma guarantees the existence of a trichromatic triangle–which, by the informal arguments above, implies that there should be a fixpoint in the interior of the triangle. If the triangle is sufficiently small, all its points are close to this fixpoint, so we have found an approximate fixpoint. A continuity argument proves that, (using precise definitions), under suitable conditions, this informal reasoning can be made into a rigorous proof.

7.6 Putting it all Together

We now have all the elements to give an outline of how an efficient algorithm for End of the Line can be used to

build an efficient algorithm to find Nash Equilibria. Given a game, we start by viewing all possible mixed strategies as points in a high-dimensional space, as outlined in Section 6. We know, from the same section, that there is a function f that at each configuration x that is not a fixpoint will give us a vector $f(x)$ x that points to a direction which yields a better outcome for some player, and if x represents a Nash Equilibrium, $f(x) = x$. We then perform the procedure of section 7: we cover the space with small polytopes, and color the vertices of the polytope according to the directions given by the vector $f(x)$ x at point x. It is important to remind the reader that we do not actually "write out" the polytope (which will have exponentially many points.) What we do is to give an algorithm, that can compute where the points are, what their neighbors are, etc. Among the polytopes constructed, Sperner's Lemma ensures that there will be one whose vertices are colored with all different colors, and we saw in subsection 7.5 that this yields a point that represents an (approximate) Nash Equilibrium.

It was important to use the constructive proof of Sperner's Lemma (Theorem 20) because the polytope that provides a solution to the problem of computing a Nash Equilibrium is found by following a path to its endpoint. More precisely, in the proof (of Theorem 20) we built a directed graph H, with vertices of indegree and outdegree at most 1, a start vertex of indegree 0 and outdegree 1, and we needed to find the vertex of outdegree 0 that is at the end of the path[16] The directed graph H can be constructed from the specification of the game whose Nash Equilibrium we seek to find. H has at most exponential size, its vertices have polynomial length names, and it can be constructed in time polynomial in the size of the game. By the last sentence, we mean "constructed" in the manner explained in Section 7.1 and the manner in which we "constructed" the polytopes: there is a polynomial size circuit that, given the

name v of a vertex of H, tells us the name of the vertex w that the outgoing edge from v leads to, or that no such vertex exists–v is the end of the line.

Finally note, that the problem described in the previous paragraph is exactly the *End of the Line* problem (see Definition 17.) So if there was an efficient (polynomial time) algorithm for the End of the Line problem, we could combine all the techniques described in this subsection and obtain a polynomial time algorithm for Nash Equilibria.

8 Nash Equilibria can Solve End of the Line

The difficulty of finding Nash Equilibria is implied by the converse of Theorem 18, namely

Theorem 21 *Given a polynomial time algorithm for (approximate) Nash Equilibrium, we can solve the* End of the Line *problem in polynomial time.*

Since we believe that there is no polynomial time algorithm for End of the Line problem, this provides us with compelling evidence that there cannot be a polynomial time algorithm for finding Nash Equilibria.

As in the proof of Theorem 18, we must exhibit a polynomial time algorithm to accomplish a difficult task, in this case, the End of the Line Problem, given that we can call a subroutine that solves another difficult task, the Nash Equilibrium problem–as long as we count the cost of each such call as taking constant time. The proof consists of exhibiting such a program.

This is a difficult, and, unfortunately, quite technical construction. I will only provide some very general guideposts of the path that leads to this algorithm, and try to illustrate some of the interesting ideas behind the techniques used.

First,we show that we can solve the End of the Line problem, given an algorithm that solves the Brouwer Fixpoint Theorem. We have seen the converse of this in Section 7.5: given a function $f()$, we built a fine triangulation of its domain, and a (Sperner) coloring such that the fixpoint could be found as a trichromatic triangle of the coloring. It is a bit tricky, but not really difficult to do the converse: given an End of the Line problem, from the directed graph H that is used in the definition of the End of the Line problem, we build a function $f()$ such that the End of the Line vertices of H (the vertices of outdegree 0) are the fixpoints of $f()$. More precisely, we can explicitly and in polynomial time obtain, from the specification of an End of the Line problem (i.e. the circuits the define the directed graph H where we want to find the "end of the line vertices"), the specification of a continuous function $f()$ such that the (approximate) fixpoint(s) of $f()$ correspond to the vertices of outdegree 0 of H. Again, by "specification" we mean that we have built a circuit that given a point x of the domain, returns $f(x)$. Moreover, this circuit is of polynomial size.[17]

Assuming that the equivalence above allows us to deal with the problem of finding fixpoints of functions, rather than directly solving the End of the Line problem, we still need a method to solve the Brouwer Fixpoint problem, given an algorithm for Nash Equilibria. Perhaps the most intriguing of the obstacles to be overcome is the transition from a static snapshot – a Nash Equilibrium – to the dynamic problem of computing a fixpoint. The computation of the fixpoint could, in principle done with the following brute force technique: compute $f(x)$ at very many points x – say at all points of a very fine mesh grid – and compare x and $f(x)$. The two quantities will be approximately equal near a fixpoint. The point x at which this happens is an approximate fixpoint–which is what we

set out to find. This strategy is not feasible, as the number of points where we have to compute $f()$ is exponentially large (it is proportional to the number of vertices of the graph H.) It is not clear *a priori* how the ability to compute Nash Equilibria can make this task become easy. We sketch the main technique below.

We may assume that $f(x)$ is computed by a circuit, consisting of 2-input *gates*. Some of the gates add their inputs, some multiply them, some compare its two inputs. The very clever trick is to *simulate each gate by a small game*, where the Nash Equilibrium of the game occurs exactly at values that correspond to correct behavior of the gate simulated. The example below should make this clearer.

Suppose we have an arithmetic multiplication gate G with two inputs u and v and output $w = u \quad v$. We may assume without loss of generality that u, v and w are all between 0 and 1.[18] Now consider a game with three players, U, V, and W . Each player has two possible (pure) strategies, say α_U and β_U for player U , α_V and β_V for player V, and α_W and β_W for player W . Now set up the payoff matrices for the game so that if U plays strategy α_U with probability u (and therefore, plays strategy β_U with probability 1 u), and V plays strategy α_V with probability v then it is a Nash Equilibrium of the 3-person game if player W plays strategy α_W with probability $w = u\ v$. In other words, the probabilities of the α strategies become a Nash Equilibrium exactly when they are the values computed by the gate.

Of course, this is not the whole story. There are tricks to make all these small games into one huge 3-person game, there are many nontrivial technical problems that must be overcome, but these have been successfully negotiated [7, 8]. Moreover, in another tour de force, the number of players was brought down from 3 to 2 [3]. The final result

reduces the End of the Line Problem to Nash Equilibria, completing the proof that computing Nash Equilibria is PPAD-complete, and, therefore, unlikely to have efficient algorithms.

9 Consequences–No Silver Bullets

The main point of the preceding sections is encapsulated in Theorem 21[19]. In plain English *there are games whose Nash Equilibria cannot be found efficiently*. Even more can be said: there are several well known algorithms in Game Theory– for example [21, 14] – that use a step by step approach to get closer and closer to a Nash Equilibrium, and provably find one. One can show, by techniques similar to the ones used to prove Theorem 21, that these algorithms must be very inefficient in general, and they may take an exponential number of steps.[20]

The approach outlined above, due to Daskalakis, Goldberg, and Papadimitriou, yields only the *PPAD*-completeness for 3-player games. A complete proof can be found in [8], and a concise technical outline in [7]. The extension to 2-player games, by Chen, Deng, and Teng, is a separate tour de force [3] – it was unexpected since, in contrast to 2-player games, it is known that there are 3-player games with rational payoff functions that only have Nash Equilibria with irrational probabilities [16]. Thus, 2-player games are somehow "simpler", and it was plausible that the lower bound for 3-person games would not extend to 2-person games.

9.1 What do these results mean for Social Scientists?

They certainly mean that, given some problem, having found a Game-Theoretic model that accurately mirrors the original situation *does not necessarily mean that a solution*

has been found. If the game is sufficiently large, we may not be able to find an actual solution. The fact that there are algorithms, like Scarf's method, that are guaranteed to solve the problem does not mean that these algorithms are efficient enough to provide us with a solution within our lifetime. Arguments that somehow imply that such algorithms represent natural mechanisms to arrive at a Nash Equilibrium may be fallacious: it may be that the mechanism of the algorithm to arrive at the equilibrium is too slow.

Such arguments generalize to other situations: the idea that actors will behave according to *rational expectations* is only realistic if they can predict the outcomes – their expectations, if you will – within reasonable time. The fact that fixpoints may be hard to compute falsifies assumptions that when systems have an equilibrium state we may assume that they have reached it: this may happen only on timescales of the age of the universe.

These negative results are yet another nail on Leibniz's optimistic expectation that rational, mathematically inspired logic will resolve human disputes. His famous quote[21] "if controversies were to arise, there would be no more need of disputation between two philosophers than between two calculators [people who compute]. For it would suffice for them to take their pencils in their hands and to sit down at the abacus, and say to each other (and if they so wish also to a friend called to help): Let us calculate" [13].

On the other hand, the existence of games whose Nash Equilibria are hard to find does not mean that finding the Nash Equilibria of a particular game is going to be hard. It may well be that it just happens to be an easy game, a very small game, or that it has some special structure that makes the problem easy. For example, it is a classical

result that for 2-player zero-sum games there is an efficient algorithm using Linear Programming – the basic mathematical ingredient is a paper by von Neumann from 1928 [17]. There are many other specific families of games, and economic models based on equilibria for which efficient, polynomial time algorithms have been found. This is an active and exciting area of research: the following short list of references is but a very small sample of the kind of results obtained [20, 4, 22, 23].

The moral of the story is that the use of Game Theory is no silver bullet that kills all vampires in the Social Sciences, or in other areas. Game Theory is a powerful tool, but often it will require more than a superficial knowledge of the main results in order for it to be useful. One should view this as excellent news: it means that there will be problems that require intelligent and possibly novel techniques, rather than mechanical application of recipes. It means that there may be opportunities for interesting collaborations between Computer Scientists, Mathematicians, and Social Scientists.

Notes

* I would thank the John Harsany Research Center and Archve for Social Science, and in particular its Director, Professor Janos Simon, for the opportunity to present a talk at the Inaugural Conference of the Institute, and for their hospitality.

1 Constantinos Daskalakis has just received the 2018 Nevanlinna Prize, in part for his work towards this result.

2 According to Chinese sources (shoushiling - „hand gesture order") was already popular during the Han Dinasty (206 BC – 220 CE)

3 160 is the length of a tweet, including signature. Accepted estimates for the diameter of the electron 10−18m, for the speed of light 3×108 m/s, for the age of the Universe 13.8×10^9 years=$13.8\times3.15\times10^{16} \approx 10$ sec

4 This is not a crippling simplification. If we want to compute a function f (x), we can ask questions of the form "Is f (x) greater or equal to a given value?". We can find the actual value with a number of questions proportional to the length of the answer. For example if we knew that the value is an integer in the range 0 to 15, we could ask first "Is f (x) ≥ 8?" If the answer is YES, we would know it is between 8 and 15. This is half of the previous range. Four more queries will reduce to possible range to 1 and yield the answer.

5 More precisely: the "normal" Sudoku game is the instance of the Valid Sudoku problem with $n = 3$. One can define the generalization for $3n \times 3n$ matrices, with entries from 1 to $3n$.

6 Actually, tthese bounds must hold only asymptotically–the precise definition is that for every algorithm there is a number $^n0$ such that $\forall n \geq n_0$ the statement holds. This is actually a challenging exercise in Algorithms. A solution

is presented in [2]. The idea behind the efficient algorithm is that a vertex with degree greater than 1 cannot be a sting, and a vertex with degree less than n − 2 (so one that is "missing" 2 edges) cannot be a body.

8 The reader may have observed that the property of being a scorpion graph is not monotone: if one adds an edge edge to the sting vertex of a scorpion graph, the resulting graph is no longer a scorpion graph.

9 A technical note: Nash's original paper [15] actually invokes Kakutani's Fixpoint Theorem.

10 The number of circuits of polynomial size is bounded by $n^{c \log n}$, as we can describe such a circuit by a list of its gates, specifying the gates where the outputs of each gate go. On the other hand, to count the number of directed graphs of 2^n vertices, with outdegree and indegree bounded by 1, consider only the graphs that are a single directed path–there are already $2^n!$ of these–and $2^n!$ is much more than $(2^n/2)^{2^{/2}}$, which is exponentially bigger than nc log n

11 See for example Sipser, Michael, Introduction to the Theory of Computation (Second ed.). PWS Publishing (2006) Section 4.2: The Halting Problem pp. 173–182

12 We use the numbers 1, 2, and 3 as the colors. If this were a computer screen we could use real colors, R(ed), B(lue), and G(reen), a notation preferred in some online references.

13 The three lines that delimit the figure do not constitute a triangle, as the lines are subdivided by vertices. They contain the edges of 7 triangles.

14 "Nice" means "continuous." The proof sketched here uses the stronger hypothesis that f satisfies an appropriate Lipschitz condition.

15 See also the remarks at the end of Section 2.1 about exact vs. approximate fixpoints.

16 Recall that the existence of such end vertex is guaranteed by Corollary 16.

17 The values x are numbers that are are n digits long, where n is the precision. Recall that both the values of the function, and fixpoints are approximate. For the mathematically inclined: there was another small difficulty to overcome. We have defined the function f only on a discrete set of points–yet we need f to be a continuos function... We extend the definition by interpolation to make the function defined in the whole domain, and this yields a continuous function.

18 Remember that f () maps the domain into itself and we can chose the domain appropriately in the beginning

19 More precisely, the result, presented informally in the Introduction, that finding Nash Equilibria is PPAD-complete

20 The precise statement is that they are PSPACE-complete.

21 quando orientur controversiae, non magis disputatione opus erit inter duos philosophus, quam inter duos computistas. Sufficiet enim calamos in manus sumere sedereque ad abacos, et sibi mutuo (accito si placet amico) dicere: calculemus.

Literature

Babai, L., Banerjee, A., Kulkarni, R., and Naik, V., Evasiveness and the Dis- tribution of Prime Numbers. STACS 2010: 71-82

Best, M.R.; van Emde Boas, P.; Lenstra, H.W. , A sharpened version of the Aanderaa-Rosenberg conjecture, Report ZW 30/74, Mathematisch Centrum Amsterdam, hdl:1887/3792 (1974).

Chen, X., Deng, X., Teng, S.-H., Settling the complexity of computing two- player Nash equilibria, J. ACM 56 (3) (2009) 1–57.

Codenotti, B., Saberi, A., Varadarajan K., and Ye, Y., The complexity of equilibria: Hardness results for economies via a correspondence with games. Theoretical Computer Science, 408 pp. 188–198 (2008).

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., Introduction to Algorithms, 3rd ed. MIT Press, Cambridge MA (2018)

Dasgupta, S., Papadimitriou, C., and Vazirani, U., Algorithms, Mc Graw Hill (2006)

Daskalakis, C., Goldberg, P.W., and Papadimitriou, C. H., The Complexity of Computing a Nash Equilibrium, Communications of the ACM, v.52, n. 2, (February 2009)

Daskalakis, C., Goldberg, P.W., and Papadimitriou, C. H., The Complexity of Computing a Nash Equilibrium. SIAM Journal on Computing v. 39 n.1: 195-259 (2009)

Harsányi, J. C., Games with incomplete information played by "Bayesian" play- ers. Part I, Management Science v.14 n. 3 (November 1967) pp. 159-182

Harsányi, J. C., Games with incomplete information played by "Bayesian" players. Part II: Bayesian Equilibrium Points, Management Science v. 14 n. 5 (January 1968) pp. 263-382

Harsányi, J. C., Games with incomplete information played by "Bayesian" play- ers. Part III: Basic Probability Distribution of the Game, Management Science v. 14 n. 7 (March 1968) pp. 383-512

Kleinberg, J., and Tardos, E., Algorithm Design, Pearson (2006)

Leibniz, G., De arte characteristica ad perficiendas scientias ratione nitentes in C. I. Gerhardt (ed.), Die philosophischen Schriften von Gottfried Wilhelm Leibniz (7 vols. 1871–1890) VII 125.

Lemke, C. E., and Howson, J. T., Equilibrium points of bimatrix games. SIAM Journal on Applied Mathematics. v. 12 n. 2: 413–423 (1964)

Nash, J. F. Jr., Equilibrium points in n-person games. Proceedings National Academy of Sciences n. 36 v.1 (January 1950) pp. 48-49

Nash, J. F. Jr., and Shapley, L. S., A Simple Three-Person Poker Game in Kuhn,

H. W. and Tucker, A. W. Contributions to the Theory of Games, Princeton U. Press, pp. 105-116 (1950)

von Neumann, J., Zur Theorie der Gesellschaftsspiele. Mathematische Annalen. 100: 295–300 (1928).

Papadimitriou, C. H., On the complexity of the parity argument and other inefficient proofs of existence, J. Computer and System Sciences v. 48 n. 3 488-532 (1994)

Rosenberg, A.L., On the time required to recognize properties of graphs: a problem SIGACT News, 5 (Oct. 1973)

Rubinstein, A., Settling the complexity of computing approximate two-player Nash equilibria. Proceedings. of the 57th FOCS, New Brunswick, USA, IEEE Press pp. 258–265 (2016)

Scarf, H., The Approximation of Fixed Points of a Continuous Mapping, SIAM Journal of Applied Mathematics, Vol.15, No. 5 (1967)

Schuldenzucker, S., Seuken, S., and Battiston, S., Finding clearing payments in financial networks with credit default swaps is PPAD-complete. Proceedings of the 8th Innovations in Theoretical Computer Science (ITCS) Conference, Berkeley, USA, (2017).

Vazirani, V. V., and Yannakakis, Y., Market equilibrium under separable, piecewise-linear, concave utilities. Journal of the ACM, v. 58 n. 3, (2011)

Paper presented at the International Memorial Conference about John C. Harsányi held place in Budapest at the 1-2 of June 2018.