# The Computational Complexity of Nash Equilibria: An Analysis of TFNP and PPAD-Completeness

Sawyer Maloney & Drew Bevington

## 1 Introduction

Since Nash proved that there is a mixed Nash equilibrium for every game [1], the computational complexity of finding that solution has seemed as intractable as the P vs NP problem itself. However, progress was made towards more precisely defining the difficulty of finding Nash equilibria within the greater context of NP problems.

In particular, we study "The Complexity of Computing a Nash Equilibrium" by Daskalakis, Goldberg and Papadimitriou [2] as well as "On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence" by Papadimitriou [3]. We report their findings as it pertains to a new complexity class, TFNP, and subclasses of particular import to the Nash problem. At the end of the paper, we also make note of other recent developments in the TFNP space and their application and impact on game theory writ large.

For specificity, we refer to the problem of finding a mixed-strategy Nash Equilibrium as NASH or FINDNASH within this paper. In particular, we are interested in an algorithm that finds a mixed-strategy Nash Equilibrium given specified payoffs for each player for every pure strategy profile.

## 2 TFNP & sub-classes

TFNP is the "Total Functional Non-Deterministic Polynomial Time" class. More specifically, it is the collection of functional problems in NP that are guaranteed to always have a solution (which is what makes them total). TFNP is a semantic class, meaning that its requirement cannot be enforced mechanically or algorithmically; instead, it is a description of how the machine (in our case, a deterministic turing machine) interacts with a given problem. As a semantic group, TFNP is unlikely to have complete problems [3]. The subgroups that will eventually be defined under TFNP will instead have verifiable and enforceable requirements for problems to be in their respective subgroups, which will mean that they are syntactic, and will have complete problems.

Papadimitriou notes that there is an asymmetry in the hardness of NP problems [3]. The non-existence of a solution is not guaranteed to be as easy as the finding of a solution, and it is often this non-existence that gives way to the intractability of the problem. Since we do not have to deal with the non-existence of a solution, the hardness of the problem must arise from some other mechanism. This absence of complexity–in the guarantee that there is a solution–places TFNP problems in a unique middle ground between NP and P. If it were proven that NP = P then it would follow that TFNP = P. However, if it were proven that NP $\neq$ P then it would not necessarily follow that TFNP $\neq$ P since TFNP is not a perfect subclass of NP. That is, we can think of the difficulty of problems in TFNP as being somewhere in the middle of P and NP. Since there are many natural problems that arise in this space, it is an interesting class of study.

In "On the complexity of the parity argument and other inefficient proofs of existence", Papadimitriou showed that, while it is unlikely that there are complete problems for TFNP, there are (at least) four subclasses of TFNP that have complete problems [3].

This requirement of having a solution is enforced through a number of exponentially non-constructive combinatorial lemma. The term 'exponentially non-constructive' means a lemma that does not explicitly give the structure, but only shows that the structure exists, and that trying to construct such a structure would be exponentially difficult. It is this non-constructive step that gives the problem its hardness for sequential computing; if it was not non-constructive, then a brute force approach would be able to solve the problem.

The most important of these lemmas are defined by Papadimitriou from the above paper:

- Parity argument: if a graph has a node of odd degree, then there must be another odd degree node. This argument creates the PPA subclass of TFNP.
- Parity for directed graphs: If a directed graph has an unbalanced node (a vertex with different in and out degrees), then there must be another unbalanced node. This creates the PPAD subclass.
- Polynomial Local Search: Every directed acyclic graph must have a sink. This argument creates the PLS subclass.
- Pigeonhole Principle: If a function maps n elements to n-1 spots, then there must be a collision. This argument creates the PPP subclass.

# 3 Intractability of Nash Equilibria

Nash's theorem states that there is always a mixed strategy Nash equilibrium for any game of finite players [1]. This proof of existence causes a number of problems for evaluating FINDNASH's complexity. The clearest result is that

FINDNASH cannot be NP-Complete. NP-Complete problems–the most difficult problems in NP–are not guaranteed to have a solution; this fact is instrumental in showing, for example, the satisfiability problem's intractability. Thus, FINDNASH is not NP-Complete which necessitates its membership to another distinct family of problems. Papadimitriou notes that we must start from the same point that led us to the P versus NP conversation in the beginning: find a class of similar problems as FINDNASH, then compare their runtimes and try to find reductions from one problem to another. In this case, FINDNASH is complete in the subclass PPAD.

To analyze FINDNASH in the context of the PPAD class, Papadimitriou introduces a the End-of-the-Line problem as the archetypal problem for PPAD; that is, PPAD as a class is defined as all problems that can reduce to End-of-the-Line efficiently (which is analogous to defining NP as all problems that efficiently reduce to the SAT problem, for example). The End-of-the-Line problem is as follows:

We are given a graph represented implicitly by Predecessor and Successor boolean circuits which run polynomially in n. We are given a source node. Our task is to find a node with no out edges.

By the PPAD lemma, we know that one such node must exist. Since we are given a node with no in edges, we know that one with no out edges must exist, and particularly it must exist at the end of the line defined by the start node. However, given the computational complexity of the boolean circuits, there does not seem to be a way to find that node without following the line, which is NP-Hard for this trivial solution.

It should be noted that we believe that there are hard problems in PPAD. Since PPAD is a subset of NP, if P = NP, then PPAD does not have hard problems (and neither does NP). However, even if P != NP, it is still possible that PPAD would have easy problems. Thus, the belief is that PPAD-Complete problems are NP-Hard, but that belief is necessarily weaker than the belief that NP-Complete problems are tractable, since the requirements for a problem to be in PPAD are strictly less than those to be in NP.

With End-of-the-Line as our PPAD-Complete example, we can now find an efficient reduction for FINDNASH.

# 4 Reduction of FINDNASH

An important characteristic of problem families is the idea of completeness. If a problem is complete this means that all other problems in the set can be reduced down to a base version which emulates the complete problem, guaranteeing that the reduced problem will have all the same characteristics of the complete problem. With a non-trivial proof it can be said that FINDNASH is a PPAD-complete problem.

The reduction and proof of FINDNASH is one that will happen in multiple steps. The first step is to prove that Brouwer's Fixed Point Problem is in PPAD by reducing it to the End-of-the-Line problem. Then, we prove that FINDNASH

can be represented by Brouwer's with FINDNASH therefore also being reducible to End-of-the-Line and thereby proving that FINDNASH is PPAD-complete.

Brouwer's Fixed Point Theorem states that any continuous map of a convex (without holes), compact (closed and bounded) space is always going to have a fixed point. If you were to have a 3d sphere and you want to move it while keeping it bounded within the confines of its original space then no matter the rotation, scaling, orientation, etc., there will always remain at least one point which stays in the same position. The search problem BROUWER gives the task of finding the fixed point if you are given a continuous function following the rules above mapped to itself. Following Papadimitriou's approach [2] we are able to represent the continuous space by a mesh of fine triangles with their vertices labeled one of 3 colors dependent on their direction of movement and calibrated such that if a triangle has three different colors on its vertices then it is an approximate fixed point of the space.

The fact that the space has been mapped to a finite number of triangles allows for this to be translated into an End-of-the-Line problem where there exists a path from the edge of the space to a "sink" inside of the space represented by a triangle with trichromatic coloring on its vertices. This reduction proves that Brouwer's is in fact PPAD.

The step to reduce FINDNASH to Brouwer's is a proof that Nash himself wrote in 1950 and it states: Suppose players have picked some [mixed] strategy. Unless this is already a Nash equilibrium then some players will want to change their strategy. Therefore, we are able to make a graph of the players movements where the movement is a change in strategy. The fixed points are the movements which are mapped to themselves which would describe a Nash equilibrium. Brouwers proves that a fixed point exists and therefore a Nash equilibrium exists. An approximate fixed point is also able to correspond to an approximate nash and therefore FINDNASH has been converted to Brouwer which has been proven to be PPAD so FINDNASH is PPAD.

It is non-trivial to conclude that FINDNASH is PPAD complete. The general steps involve proving that End-of-the-Line is able to be translated into a corresponding game. Then, the approximate Nash equilibrium is translated into a corresponding End-of-the-Line game and finally, the graph of the game is translated into a Brouwer's space which simplifies into a 3 person game proving that FINDNASH is in fact PPAD-complete.

# 5 Impact of a P != NP on Nash

While widely believed, it has not yet been proven that P != NP. If it were to somehow be proven that P = NP then a wide swath of hitherto difficult problems would have an efficient solution. However, if it were to be proved that P != NP then it would be guaranteed that NP problems are intractable. TFNP problems could still be easy, as they are not NP-Complete. It is widely believed that PPAD, with problems such as NASH, BROUWER, and End-of-the-Line contains hard problems, but it is also not guaranteed. Therefore, it is possible

that NASH is in fact $\in$ P even if NP is not.

# 6 Efficient Computation of Nash Equilibria

In a real world setting, there is a benefit to simple strategies. It is possible to imagine that a rational and self-interested agent would both prefer optimal strategies generally and choose a sub-optimal strategy given that it was sufficiently simpler than an optimal strategy. In [4], researchers present a quasi-polynomial algorithm ($n^{O(\ln(n))}$) that finds simple strategies. They use a probabilistic argument to verify that such a small Nash equilibrium exists by saying that for any multiset over the pure strategies of the two players' pure strategies, the probability that it is a "good" selection is greater than zero. Then, they use an exhaustive search algorithm to find such a Nash equilibrium once the game is defined. The equilibria that are found through this method are approximate Nash equilibria, and are almost as good as strict Nash equilibria in many applied settings [4]. According to [4], this is the first subexponential algorithm for Nash equilibria, which has a number of important impacts. First, since the strategies themselves are simple, it stands to reason that they would be more attractive for a rational but finite agent who is looking for a strategy for the game. Also, since the problem is relatively more tractable then finding a pure Nash equilibrium, it means that a normal agent would also be significantly more likely to be able to find said equiblirium in the course of normal play.

# 7 Subclasses of TFNP

While we outlined and focused on the time complexity and PPAD-completeness of Nash, there are constant innovations in the field of TFNP as a whole. TFNP is a problem space proven to be volatile in its constant addition and collapse of an ever-increasing family of sub-classes. An example of this is in the work of Alexandros Hollender with a reduction of Gradient Descent in the CLS (Continuous Local Search) space to a problem which proves that CLS = PPAD $\cap$ PLS. The problem of Gradient Descent is one whose applicability to fields like machine learning has catapulted itself to the forefront of TFNP research bringing more attention to the problem family as a whole. The Gradient Descent algorithm is also a prime example of a natural problem that arises in the TFNP space, which lends credence to the idea that the space as a whole is important. Particularly for proofs that create new groupings of complexity classes, finding natural problems is important to illustrate the possible types of problems that would fall under the class.

# References

[1] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[2] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a Nash equilibrium," in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pp. 71–78, ACM, 2006.

[3] C. H. Papadimitriou, "On the complexity of the parity argument and other inefficient proofs of existence," *Journal of Computer and System Sciences*, vol. 48, no. 3, pp. 498–532, 1994.

[4] R. J. Lipton, E. Markakis, and A. Mehta, "Playing large games using simple strategies," in *Proceedings of the 4th ACM Conference on Electronic Commerce*, EC '03, pp. 36–41, ACM, 2003.