Game Theory

Lecture Notes By

Y. Narahari

Department of Computer Science and Automation

Indian Institute of Science

Bangalore, India

August 2012

Complexity of Computing Nash Equilibrium

Note: *This is a only a draft version, so there could be flaws. If you find any errors, please do send email to* `hari@csa.iisc.ernet.in`. *A more thorough version would be available soon in this space.*

One of the central questions that theoretical computer scientists have investigated in recent times with much interest and intensity is the issue of complexity of computing a (mixed strategy) Nash equilibrium of a finite strategic form game. Thanks to the Nash theorem at least one Nash equilibrium is guaranteed to exist for finite strategic form games and the problem of finding a Nash equilibrium therefore belongs to the class of total search problems (search problems where a solution is guaranteed to exist). A large body of literature exists on this problem and recently, Daskalakis, Goldberg, and Papadimitriou [1] have written a survey paper on this topic. This chapter attempts to bring out some salient aspects of the main results. The contents of this chapter are called out from many papers in the bibliography, in particular, from [1].

1 Problems: NASH, BROUWER

First we introduce the two important problems NASH and BROUWER which are both total search problems.

- NASH: Given a finite strategic form game, find a mixed strategy Nash equilibrium of the game. There could be potentially a number of mixed strategy Nash equilibria but we are interested in only one sample equilibrium.
- BROUWER: Given a continuous function f on the set $[0,1]^m$ (which is a compact and convex set), find a fixed point of the function f, that is, find an $x \in [0,1]^m$ such that $f(x) = x$. Note that m is a finite positive integer here and potentially f may have multiple fixed points. We are interested in only finding one of those.

Instead of an exact Nash equilibrium or an exact fixed point, it is often convenient to work with an approximate Nash equilibrium or an approximate fixed point. Given a real number $\epsilon > 0$ (howsoever small), an ϵ-Nash equilibrium is a profile of mixed strategies where any player can only improve her expected payoff by at most ϵ by switching to any other strategy. Similarly, an ϵ-fixed point of a

function $f : [0,1]^m \rightarrow [0,1]^m$ is a point $x \in [0,1]^m$ such that

$$d(f(x), x) \geq \epsilon$$

where d is a metric (or distance function) defined on $[0,1]^m$. We now state two problems ϵ-NASH and ϵ-BROUWER as follows.

- ϵ-NASH: Given a finite strategic form game and a real number $\epsilon > 0$, compute an ϵ-Nash equilibrium of the game.
- ϵ-BROUWER: Given an efficient algorithm to evaluate a continuous function $f : [0,1]^m \rightarrow [0,1]^m$, a desired accuracy $\epsilon > 0$, and a distance metric d on $[0,1]^m$, compute a point $x \in [0,1]^m$ such that $d(f(x), x) \leq \epsilon$.

In the above definitions, a few technical subtleties have been left out for ease of presentation and the interested reader is referred to [1] for more details. Clearly, the problem ϵ-NASH (ϵ-BROUWER) is no more complex than NASH (BROUWER) which involves computing an exact value. Therefore any hardness result for ϵ-NASH (ϵ-BROUWER) will automatically carry over to NASH (BROUWER). From now on, for ease of presentation, we will refer to ϵ-NASH (ϵ-BROUWER) as NASH (BROUWER).

The main result presented in this chapter is that the problem NASH is PPAD-complete where PPAD is a complexity class of total search problems named after "Polynomial Parity Argument for Directed graphs:" We now describe this class PPAD and a problem called EOL (End of Line Problem) which is a PPAD-complete problem.

2 The class PPAD

2.1 The classes P, NP, NPC

First we recall the classes $\mathbb{P}$ and $\mathbb{NP}$. $\mathbb{P}$ is the class of all decision problems or search problems which are solvable in polynomial time by a deterministic Turing machine. $\mathbb{NP}$ is the class of all problems solvable in polynomial time by a non-deterministic Turing machine. $\mathbb{NP}$ can also be described as the class of all search problems of the form: "given an input, find a solution that can be verified in polynomial time (by a deterministic Turing machine) or report that a solution does not exist." We say a problem X is $\mathbb{NP}$-hard if every problem belonging to $\mathbb{NP}$ can be reduced to the problem X. Reduction of a problem Y to a problem X means that there is an efficient algorithm (that is, a polynomial time algorithm on a deterministic Turing machine) that takes as input an instance of Y and outputs an instance of X so that any solution to the instance of X can be transformed into a solution of Y using an efficient algorithm. Finally, an $\mathbb{NP}$-complete problem X is a problem which is not only -hard but also belongs to NP. In order to show that a problem Y is $\mathbb{NP}$-complete, it is enough to show that an $\mathbb{NP}$-complete problem X can be reduced to the problem Y. Also, it is useful to observe that the class $\mathbb{NP}$ consists of all problems which can be reduced to any $\mathbb{NP}$-complete problem.

2.2 The classes TFNP and PPAD

Papadimitriou [**?**] proposed the complexity class TFNP (Total Function Non-deterministic Polynomial time) to describe all search problems for which every instance has a solution. An example would be FACTOR which takes as input an integer and determines all its prime factors. An immediate relevant example is NASH which finds an exact or ϵ-Nash equilibrium of a finite strategic form game.

Papadimitriou also proposed a classification of total search problems based on certain "arguments" which are non-constructive steps that are used in proving that every instance of the total search problem has a solution. This leads to the following classes of total search problems.

- PPA (Polynomial Parity Argument): If a graph has a node of odd degree, then it must have at least one other node of odd degree. This is called the parity argument (PA) and the class of total search problems which use this argument are grouped under the class PPA.
- PPAD (Polynomial Parity Arguments for Directed Graphs): Given a directed graph, the in-degree (out-degree) of a node is the number of incoming (outgoing) arcs. A node is said to be unbalanced if its in-degree and out-degree are different. The PAD argument says that if a directed graph has an unbalanced node, then there must exist at least one other unbalanced node.
- PLS (Polynomial Local Search): In a directed graph, a sink is a node that does not have any outgoing arcs. The LS argument says that a directed graph must have a sink.

The relationship between the various classes of total search problems is depicted in Figure 1. (The figure nonchalantly assumes that $\mathbb{P} \subset \mathbb{NP}$. If $\mathbb{P} = \mathbb{NP}$, then all the classes will collapse into one).

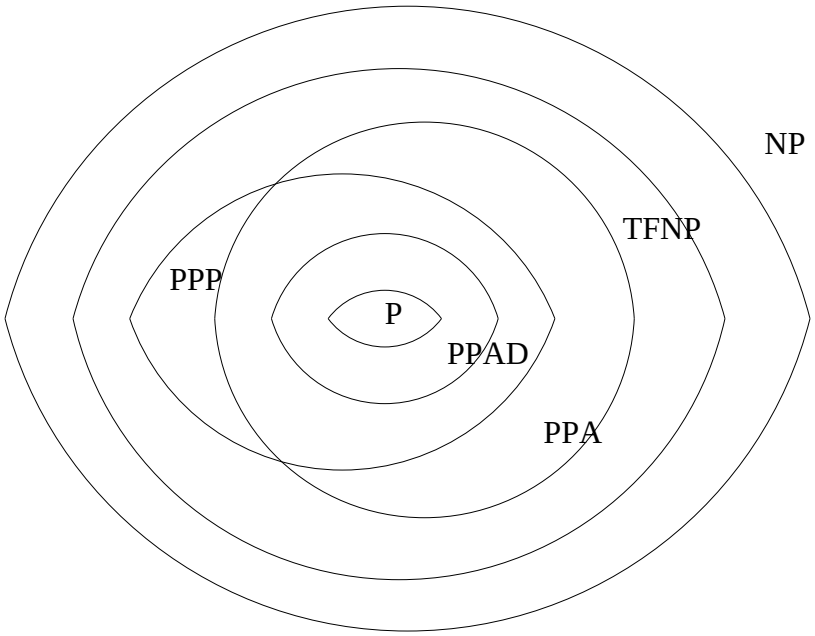

Figure 1:

Figure 1: Relationships among different relevant complexity classes.

It is interesting to think about how hard the problems in PPAD are. If $\mathbb{P} = \mathbb{NP}$, then this issue is settled since PPAD which is a subset of NP will itself be equal to P. If $\mathbb{P} \neq \mathbb{NP}$ (as is widely believed), then there is strong evidence to believe that PPAD contains hard problems. For several decades now, theoretical computer scientists have tried in vain to design efficient algorithms for some problems which are in PPAD (for example, BROUWER, NASH, and the End-of-Line problem which

will be introduced in the next section). Thus unless $\mathbb{P} = \mathbb{NP}$, we can safely believe that PPAD does contain computationally hard problems.

2.3 The EOL Problem

The EOL (End-of-Line) problem is the best known PPAD-complete problem. It is a special case of the following total search problem: Given a directed graph G and a designated unbalanced vertex of G, find some other unbalanced vertex in G. The EOL problem assumes that every vertex G has at most one incoming edge and at most one outgoing edge. With this restriction, the given graph must be a set of paths and cycles. Figure 2 provides a few representative examples of such graphs.

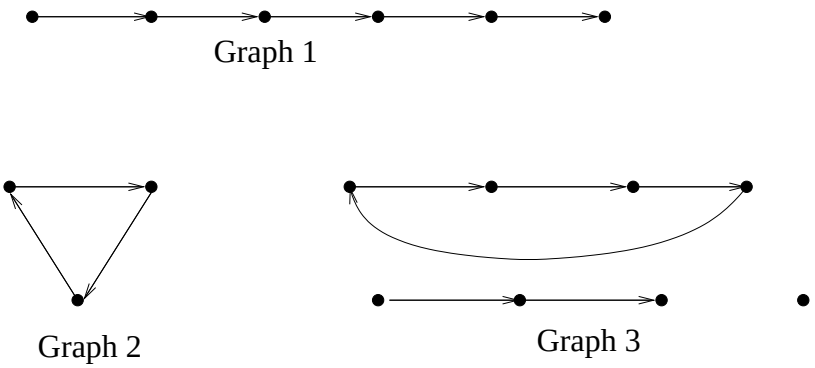

Figure 2: Some examples of the EOL problem.

The EOL problem further assumes that each vertex in a graph with, say 2^n vertices, is labeled with a unique binary string of length n. The edges in the graph are described using a predecessor function p and a successor function s defined on the set of vertices. If (v, v') is an edge in the graph, then $s(v) = v'$ and $p(v') = v$ have the obvious meaning. Note that the entire graph is represented using binary strings as vertices and by defining the p and s functions. With this representation, the EOL problem starts with a designated unbalanced vertex and seeks to find some other unbalanced vertex.

It has been shown that the EOL problem is PPAD-complete. Two immediate consequences of this fact are the following:

- A problem X is PPAD-complete if EOL can be reduced to the problem X.
- A problem $Y \in PPAD$ if Y can be reduced to EOL. In fact, PPAD is precisely the class of all problems which can be reduced to EOL.

One of the major results proved recently times is that the problem NASH is PPAD-complete. The proof of this result has taken several years of effort (see the bibliography) and the final steps in the proof was contributed by Daskalakis in his doctoral dissertation which won the ACM Doctoral Dissertation Award in 2008.

3 NASH is PPAD-complete

We only provide an outline of the proof of this result. The proof proceeds in two stages.

- First it is shown that NASH $\in$ PPAD. This is shown by reducing NASH to EOL.
- Second, it is shown that NASH is PPAD-complete. This is shown by reducing EOL to NASH.

The proof in both directions above uses the problem BROUWER.

3.1 NASH Belongs to PPAD

It has been shown that BROUWER can be reduced to EOL using the Sperner's lemma. We refer the reader to [1] for details. Therefore, BROUWER $\in$ PPAD. It has also been shown that NASH can be reduced to BROUWER (such a reduction, as pointed out by [1] was suggested by Nash himself in his 1950 paper [2]). Since NASH can be reduced to EOL, we have that NASH can be reduced to EOL which means that NASH $\in$ PPAD.

3.2 PPAD to NASH

First it is shown that BROUWER is PPAD-complete. This is done by showing that EOL can be reduced to BROUWER by encoding a directed graph in terms of a continuous, easy-to-compute function. BROUWER can be reduced to NASH using interesting arguments [1]. Since BROUWER is PPAD-complete and NASH can be reduced to BROUWER, we have that NASH is PPAD-complete.

For detailed proofs of all the above results, the reader is referred to [1] and other relevant papers in the bibliography.

3.3 Some observations

The fact that NASH is PPAD-complete and therefore computationally a very hard problem (unless $\mathbb{P} = \mathbb{NP}$) raises some questions about whether it is practical to use it as a solution concept. Efficient computation of Nash equilibrium is crucial for many reasons [3].

- In many practical situations such as game playing, multi-agent reasoning, auctions, etc., quick computation of equilibrium is extremely useful.
- If players in a game can only invest polynomial computation in a game, then it is important that Nash equilibrium or any chosen solution concept is computable in polynomial time.

The result that NASH is PPAD-complete almost rules out the possibility of a polynomial time algorithm for computing Nash equilibrium. Notwithstanding this, Nash equilibrium will retain its position as a premier solution concept due to its intuitive, natural, and technically sound development. We now summarize a few important results relevant to computation of Nash equilibrium.

- Though NASH is PPAD-complete, it is unlikely to be $\mathbb{NP}$-complete, as argued by Nimrod Megiddo [1]: Suppose we have a reduction from the $\mathbb{NP}$-complete problem SAT to NASH. Since NASH is a total search problem and SAT is not a total search problem, the above reduction would mean that the solution of any instance of NASH tells us whether or not the SAT instance has a solution. This enables us to design a non-deterministic algorithm to verify that an instance of SAT has no solution. The existence of such an algorithm for SAT is considered by complexity theorists as much unlikely as $\mathbb{NP} = \mathbb{NP}$.

- Chen, Deng, and Teng [4] have ruled out the existence of a fully polynomial time approximation scheme (FPTAS) for computing an ϵ-Nash equilibrium for finite strategic form games. This means there is no algorithm that can compute an ϵ-Nash equilibrium with a running time that is polynomial in the size of the game (number of players and number of strategies) and polynomial in $1/\epsilon$.

- On the other hand, it is open whether thee exists a polynomial time approximation scheme (PTAS) for computing an ϵ-Nash equilibrium (that is an algorithm that computes an ϵ-Nash equilibrium with running time that is polynomial in the size of the game but depends arbitrarily on $1/\epsilon$).

- Lipton, Markakis, and Mehta [5] have shown the existence of a sub-exponential algorithm for computing an ϵ-Nash equilibrium. The complexity of their algorithm is

$$O\left(k \log \frac{k}{\epsilon^2}\right)$$

  where k is the size of the game that captures the number of players and the sizes of the strategy sets. This is a small ray of hope in the midst of a sea of negative results.

- It is worth recalling here that the complexity of computing Nash equilibria in matrix games (two player zero-sum game) is polynomial, thanks to the linear programming formulation of von Neumann and Morgenstern. The complexity of this problem for two player non-zero sum games and beyond is unlikely to be polynomial time unless we are looking at very special cases. Numerous algorithms have been proposed over the last 50 years, however these algorithms either have worst case exponential time complexity or their complexity is not even known. For a survey of these algorithms, the reader is referred to McKelvey and McLennan [6]. Chapter 10 has already provided some relevant details.

4 To Probe Further

Much of the content in this chapter owes to the excellent survey paper by Daskalakis, Goldberg, and Papadimitriou [1]. The detailed proofs of results are available in [7, 1, 4, 8]. There are many other survey articles on both complexity and algorithmic issues in the volume on Algorithmic Game Theory edited by Nisan, Roughgarden, papers by Conitzer and Sandholm [9] and Roughgarden [3] are also expository. There is a good discussion in the book by Shoham and Leyton Brown [10]. Other useful references include [5, 11].

References

[1] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. *Communications of the ACM*, 52(2):89–97, 2009.

[2] John F. Nash Jr. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36:48–49, 1950.

[3] T. Roughgarden. Algorithmic game theory. *Communications of the ACM*, 53(7):78–86, 2010.

[4] X. Chen, X. Deng, and S. H. Teng. Computing nash equilibria: Approximation and smoothed complexity. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 603–612. IEEE, 2006.

[5] R. J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 36–41. ACM, 2003.

[6] R. D. McKelvey and A. McLennan. Computation of equilibria in finite games. *Handbook of Computational Economics*, 1:87–142, 1996.

[7] C. Daskalakis. The complexity of computing a nash equilibrium. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 71–78. ACM, 2006.

[8] X. Chen and X. Deng. Settling the complexity of two-player nash equilibrium. *Journal of ACM*, 56(3), 2009.

[9] Vincent Conitzer and Tuomas Sandholm. Complexity results about nash equilibria. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico*, pages 765–771, 2003.

[10] Y. Shoham and K. Leyton-Brown. *Multiagent systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, New York, USA, 2009, 2009.

[11] K. Etessami and M. Yannakakis. On the complexity of nash equilibria and other fixed points. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 113–123. IEEE, 2007.