6.891 Games, Decision, and Computation February 19, 2015

Lecture 4

Lecturer: Constantinos Daskalakis *Scribe: Constantinos Daskalakis*

1 Overview

We have previously discussed games, rationality, knowledge, and solution concepts. We also talked about mechanism design – where the goal is to engineer a game that, when played by rational players, will induce a desired outcome (and the set of desired outcomes may depend on information known only to the players, and not to the designer). We also talked about existence theorems for Nash equilibria, including Nash's theorem for general games (using Brouwer's fixed point theorem) and von Neumenn's theorem for two-player zero-sum games (via linear programming duality).

In this lecture, we will introduce algorithms for finding a Nash equilibrium in games other than just two-player zero-sum games. We will investigate how computationally complex it is to find the Nash equilibrium of a game (for the game theorist and for the players themselves). We will first discuss support enumeration algorithms, then we will discuss algorithms for symmetric games, and we will end with a discussion of the Lenke-Howson algorithm.

2 Support Enumeration Algorithms

In this section, we will consider whether knowing the *support* of a Nash equilibrium can reduce the computational complexity of solving for a Nash equilibrium. We define the *support* of a mixed strategy x to be the set of pure strategies with strictly positive probability mass in x.

2.1 Two-player games

Suppose we have an m by n two-player game (R, C), and let S_R, S_C be the supports of the Row and Column players' mixed strategies, respectively, at some Nash equilibruim of the game. Using this information, we can construct the following linear program to find a Nash equilibrium (x, y):

$$\begin{aligned} & \max 1 \\ \text{s.t. } & e_i^T R y \geq e_j^T R y, \forall i \in S_R, \forall j \in [m] \\ & x^T C e_i \geq x^T C e_j, \forall i \in S_C, \forall j \in [n] \\ & \sum x_i = 1 \text{ and } \sum y_i = 1 \\ & x_i = 0, \forall i \notin S_R \text{ and } y_j = 0, \forall j \notin S_C \\ & x_i \geq 0, \forall i \text{ and } y_j \geq 0, \forall j \end{aligned}$$

Hence, we can solve for a Nash equilibrium using the above LP, if we already know the support of the equilibrium. If we don't already know the support, it is reasonable to consider solving for a Nash equilibrium by repeatedly guessing the support and then solving the resulting LP. Using this algorithm to solve for a Nash equilibrium, we get a runtime of $2^{m+n} \cdot \text{poly}(|R|, |C|)$. The 2^{m+n} factor is the number of possible supports we might need to guess, and for each support that we guess, we can solve the LP in a runtime of $\text{poly}(|R|, |C|)$.

Corollary 1. *In any two-player game, there exists a Nash equilibrium whose mixed strategies use only rational numbers in their probability distributions.*

Proof: This follows from the correctness of the support enumeration algorithm. If there is a Nash equilibrium with supports S_R and S_C, then the polytope of the corresponding LP is non-empty. Any vertex of that polytope is a Nash equilibrium whose coordinates are rational and whose bit complexity is polynomial in the description of the LP, and hence the description of the game. □

This corollary suggests that finding an exact, as opposed to an approximate, Nash equilibrium in polynomial-time is not obviously infeasible. However, the corollary is not true for k-player games where $k > 2$. Indeed, Nash's original paper [3] contains an example of a 3-player game with only irrational equilibria.

2.2 n-player games

Now, let's consider the case of an n-player game for arbitrary n. Even if we know the support of each player (denoted Σ_p for player p) in a Nash equilibrium, the feasibility problem we need to solve, shown below, is unfortunately not an LP:

$$\begin{aligned} \forall p: \quad & u_p(e_i; x_{-p}) \geq u_p(e_j; x_{-p}), \forall i \in \Sigma_p, \forall j \in S_p; \\ & \sum_{i \in S_p} x_p(i) = 1; \\ & x_p(i) = 0, \forall i \in S_p \setminus \Sigma_p; \\ & x_p(i) \geq 0, \forall i \in S_p. \end{aligned}$$

Note that the expected payoff $u_p(e_i; x_{-p})$ in the above problem is a polynomial of degree $n-1$ in the variables x_{11}, x_{12},..., x_{nk} (let's say that all players have strategies $\{1, \ldots, k\}$). This means that we need to solve a system of polynomial equations and inequalities in $n \cdot k$ variables, where the polynomials have degree $n-1$. This can be done to precision ϵ in time polynomial in $n^{nk} \log(\frac{1}{\epsilon})$, using tools from the existential theory of the reals. Thus, including an extra factor of 2^{nk} for guessing the support of the Nash equilibrium, we get total running time polynomial in $n^{nk} \log(\frac{1}{\epsilon})$. Recalling that the description of the game already needs to specify $nk^n = n2^{n \log(k)}$ numbers, the running time is quasi-polynomial as long as k scales polynomially with n.[1]

if discretize $\frac{n}{\epsilon}$, runtime $(n/\epsilon)^{nk}$

3 Algorithms for Symmetric Games

We will now discuss whether the running time for finding a Nash equilibrium can be improved for the class of games called *symmetric games*.

Definition 1. *An n-player game is symmetric iff:*

- *All players have the same strategy set: $S = \{1, ..., k\}$; and*
- *there exists some function g of $k+1$ arguments such that every player's utility can be written as: $u_p(s) = g(s_p; n_1(s_{-p}), ..., n_k(s_{-p}))$, where $n_i(s_{-p})$ is the number of players other than p choosing strategy i in pure strategy profile s.*

For example, rock-paper-scissors and guess $\frac{2}{3}$ of the average are both symmetric games. Also, congestion games, where players choose routes between the same source and destination, are symmetric games. Notice that describing symmetric games can be done much more succinctly than general games. In particular, a n-player k-strategy symmetric game can be described by specifying $O(\min\{kn^{k-1}, k^n\})$ payoffs, which saves exponentially in description complexity when n is large and k is small.

[1] A *quasi-polynomial-time algorithm* for some computational task is an algorithm that solves an instance Π of the task in time $2^{\text{poly}(\log d(\Pi))}$, where $d(\Pi)$ is the description complexity of instance Π. If the polynomial in the exponent of the running time is of degree 1 the algorithm is called *polynomial-time*.

3.1 Symmetric equilibria

Definition 2. *A Nash equilibrium $x \in \times\Delta_p$ is called symmetric if $x_1 = x_2 = \ldots = x_n$.*

Theorem 1 (Nash '51 [3])**.** *In every symmetric game, there exists a symmetric Nash equilibrium.*

Proof: Recall Nash's function $f : \times_p \Delta_p \to \times_p \Delta_p$, which maps some x to a y defined as follows, for all players p and strategies $j \in S_p$:

$$y_p(j) = \frac{x_p(j) + \max(0, u_p(j; x_{-p}) - u_p(x))}{1 + \sum_{j \in S_p} \max(0, u_p(j; x_{-p}) - u_p(x))}.$$

Suppose we restrict the domain of Nash's function to the set:

$$\times_p \Delta_p \cap \{x_1 = x_2 = ... = x_n\}.$$

Then the range of the function will be a subset of this same restricted set, since every player performs the same "update" in the function f. So f maps points of the restricted set to points in the same set. Moreover, the set is convex, closed and bounded. So we can use Brouwer's fixed point theorem to show the existence of a fixed point in the restricted set. This fixed point is a Nash equilibrium as we saw in the proof of Nash's theorem in Lecture 2. And since it belongs to the restricted set, it must be a symmetric one. □

A symmetric Nash equilibrium x of a n-player k-strategy symmetric game can be found as follows:

- Guess the support of x: 2^k possibilities;
- Write down a system of polynomial equations and inequalities corresponding to the equilibrium conditions for the guessed support. (This is a simplified version of the system we wrote down for general games in Section 2.2.) The polynomials involved have degree $\leq n - 1$ in k variables (c.f. $k \cdot n$ for general games), so the system can be approximately solved to precision ϵ in time polynomial in $n^k \log(\frac{1}{\epsilon})$, using tools from the existential theory of the reals.

The overall running time is polynomial in $n^k \log(\frac{1}{\epsilon})$, which is polynomial in the size of the input for $k = n^{1-\delta}$, for any constant $\delta > 0$.

3.2 Symmetrization

We will show a reduction from the problem of finding a Nash equilibrium of a two-player game to the problem of finding an equilibrium of a symmetric two-player game. The reduction we present is due to Gale, Kuhn and Tucker [1].

Suppose that we are given an arbitrary two-player game $\mathcal{G}_1 := (R, C)$, and we want to find a Nash equilibrium of this game. WLOG, suppose that R and C have only positive entries.

Exercise 1. *Show that computing a Nash equilibrium of an arbitrary game $\mathcal{G}$ can be reduced to the problem of computing a Nash equilibrium of a game $\mathcal{G}'$ whose entries are positive.*

The reduction constructs a $2n \times 2n$ symmetric game $\mathcal{G}_2$, with the following payoff matrices in block form:

$$\begin{pmatrix} 0, 0 & R, C \\ C^T, R^T & 0, 0 \end{pmatrix}.$$

Theorem 2. *Given a Nash equilibrium of $\mathcal{G}_2$, we can efficiently compute a Nash equilibrium of $\mathcal{G}_1$.*

Proof: Suppose we are given a Nash equilibrium of $\mathcal{G}_2$. We denote this equilibrium by $([x_1; y_1], [x_2; y_2])$, where $[x_1; y_1]$ is the mixed strategy of the row player and $[x_2; y_2]$ the mixed strategy of the column player in block form, as shown in the following diagram:

$$\begin{array}{c} \\ x_1 \\ y_1 \end{array} \begin{array}{c} \begin{array}{cc} x_2 & y_2 \end{array} \\ \begin{pmatrix} 0,0 & R,C \\ C^T,R^T & 0,0 \end{pmatrix} \end{array}.$$

First, at least one of x_1 and y_1 must be nonzero. Assume WLOG that $x_1 \neq 0$. We claim the following:

Claim 1. *$x_1 \neq 0$ implies that $y_2 \neq 0$.*

Proof: Using our positivity assumption for the entries of the game, if $y_2 = 0$, then the row player in game $\mathcal{G}_2$ could improve his expected payoff by setting $x_1 = 0$ and boosting all probabilities in y_1 by a factor of $1/|y_1|_1$. This contradicts that $([x_1; y_1], [x_2; y_2])$ is a Nash equilibrium. □

Claim 2. *Let $\hat{x_1} = \frac{x_1}{|x_1|_1}$ and $\hat{y_2} = \frac{y_2}{|y_2|_1}$. Then $(\hat{x_1}, \hat{y_2})$ is a Nash equilibrium of (R, C).*

Proof: By contradiction. Suppose $(\hat{x_1}, \hat{y_2})$ is not a Nash equilibrium of $\mathcal{G}_1$. Then WLOG the row player can improve his expected payoff by switching to some $\tilde{x_1}$. Then

$$\tilde{x_1}^T R \hat{y_2} > \hat{x_1}^T R \hat{y_2}. \qquad (*)$$

Subclaim 1. *(*) implies that the row player in $\mathcal{G}_2$ can improve his payoff by switching to $[\tilde{x_1} \cdot |x_1|_1; y_1]$ from $[\hat{x_1} \cdot |x_1|_1; y_1]$.*

Proof of Subclaim 1: The expected payoff of the row player in $\mathcal{G}_2$ from mixed strategy $[\tilde{x_1} \cdot |x_1|_1; y_1]$ is

$$|x_1|_1 \cdot \tilde{x_1}^T R y_2 + y_1^T C^T x_2.$$

His expected payoff from the mixed strategy $[\hat{x_1} \cdot |x_1|_1; y_1]$ is

$$|x_1|_1 \cdot \hat{x_1}^T R y_2 + y_1^T C^T x_2.$$

The first of these payoffs is strictly greater, due to (*). This concludes the proof of the subclaim. □
Given Subclaim 1, we get a contradiction to our assumption that $([x_1; y_1], [x_2; y_2])$ is a Nash equilibrium of $\mathcal{G}_2$. □ □

By virtue of our reduction, we conclude that it is essentially as hard to solve a symmetric two-player game as it is to solve a general two-player game. An interesting open problem is whether the symmetrization can be generalized to games with more than 2 players.

Open Problem 1. *Is there a reduction from 3-player games to symmetric 3-player games?*

4 The Lemke-Howson Algorithm

We now describe the Lemke-Howson algorithm [2], which computes an exact Nash equilibrium of any given two-player game (R, C). Since there always exists a rational equilibrium, this task is feasible.

To simplify our discussion, we assume that the input game is a symmetric $n \times n$ game, i.e. $C = R^T$. From our work in Section 3.2, we can make this assumption WLOG.

The idea of the algorithm is to perform pivoting steps between the vertices of a polytope related to the game until a Nash equilibrium is found. The (n-dimensional) polytope of interest is given by

$$R \cdot z \leq 1, z \geq 0,$$

where z is an n-dimensional vector.

We now make an additional assumption: at every vertex of the polytope, exactly n out of the $2n$ inequalities are tight. We can make this assumption because if it is not true, we can perturb the original game entries with exponentially small noise to make it happen. The equilibria of the perturbed game will be approximate equilibria of the original game, and these can be converted to exact equilibria. This is what the following exercise asks you to do:

Exercise 2. *Part 1. Show that the Lemke-Howson polytope can be perturbed with exponentially small noise so that at every vertex exactly n of the inequalities are tight. Part 2. Show that, given a Nash equilibrium of the perturbed game, an equilibrium of the original game can be recovered in polynomial time.*

Definition 3. *Pure strategy i is* represented *at a vertex z of the polytope if at least one of the following inequalities is tight:*

$$z_i \geq 0$$

$$R_i z \leq 1$$

Furthermore, we call any vertex of the polytope where all pure strategies are represented a democracy.

Notice that $(0, 0, \ldots, 0)$ is a democracy according to our definition. We make an interesting observation about democracies.

Lemma 1. *If a vertex $z \neq 0$ of the polytope is a democracy, then $(\frac{z}{|z|_1}, \frac{z}{|z|_1})$ is a Nash equilibrium.*

Proof: At a democracy we have the following implication:

$$\forall i: \quad z_i > 0 \implies R_i z = 1.$$

Hence

$$\forall i: \quad z_i > 0 \implies R_i z \geq R_j z, \forall j$$

and after normalization:

$$\forall i: \quad \frac{1}{|z|_1} \cdot z_i > 0 \implies e_i^T R \cdot \frac{z}{|z|_1} \geq e_j^T R \cdot \frac{z}{|z|_1}, \forall j.$$

Notice that these are exactly the equilibrium conditions for $(\frac{z}{|z|_1}, \frac{z}{|z|_1})$ to be a symmetric Nash equilibrium of the game. □

The goal of the Lemke-Howson algorithm is to find a democracy in the given polytope. The algorithm operates as follows. Let's call n the "special strategy," albeit this choice is arbitrary.

1. Start at the vertex $v_0 := (0, 0, \ldots, 0)$.
2. *Comment:* By non-degeneracy, there are exactly n edges of the polytope adjacent to v_0. Each of these edges corresponds to un-tightening one of the $z_i \geq 0$ inequalities which are tight at v_0.
3. Keeping all other inequalities tight, un-tighten the inequality $z_n \geq 0$ (which corresponds to our special strategy n). This defines an edge of the polytope adjacent to v_0. Go to the other endpoint of this edge.
4. If the obtained vertex v_1 is a democracy, then a Nash equilibrium has been found because $v_1 \neq 0$.
5. Otherwise, one of the strategies $1, \ldots, n-1$, say strategy j_1, is represented twice, by both $z_{j_1} = 0$ (which was already tight) and $R_{j_1} z = 1$ (which just became tight).
6. *Comment:* For the next step, we will un-tighten one of the two inequalities that are tight for j_1. If we un-tighten $R_{j_1} z \leq 1$, this would define the same edge $(v_0 v_1)$ that brought us to v_1. To make progress we will un-tighten instead the other inequality representing strategy j_1.
7. Un-tightening $z_{j_1} \geq 0$, this defines an edge $(v_1 v_2) \neq (v_0 v_1)$ of the polytope. Go to vertex v_2.
8. If v_2 is a democracy, then stop.
   Comment: To be shown that $v_2 \neq 0$, and hence $v_2/|v_2|_1$ is a symmetric Nash equilibrium.

9. Otherwise, again some strategy j_2 is doubly represented, all other strategies in $\{1, \ldots, n-1\}$ are represented once, and the special strategy n is not represented at all.

   /*Proof: *This is because, for all strategies who were singly represented before the step was taken, their corresponding inequalities are maintained tight during the step. So they are still represented. Strategy j_1 was doubly represented at vertex v_1 and we only un-tightened one of its tight inequalities. So it is still represented at vertex v_2 by the inequality that we did not un-tighten. Finally, strategy n was not represented before the step and since v_2 is not a democracy it is still not represented.*/*

10. ...

11. At step t of the algorithm:

    - if vertex v_t is not a democracy then one strategy j_t is represented twice, all other strategies in $\{1, \ldots, n-1\}$ are represented once, and strategy n is not represented at all.
    - between $R_{j_t} z \leq 1$ and $z_{j_t} \geq 0$, un-tighten the one that defines an edge $(v_t v_{t+1}) \neq (v_{t-1} v_t)$.
    - If v_{t+1} is a democracy, stop. *Comment:* To be shown that $v_{t+1} \neq 0$.
    - O.w. continue.

Theorem 3. *The Lemke-Howson algorithm terminates at a democracy that is different than* $(0, \ldots, 0)$.

Proof: The Lemke-Howson algorithm defines a walk $v_0, \ldots, v_t, \ldots$ on the vertices of the polytope. We show that the vertices encountered in this walk satisfy a special property.

Claim 3. *For all t, v_t is either a democracy, or it satisfies the following property:*

Π : *All of the strategies in $\{1, \ldots, n-1\}$ are represented at v_t, exactly one of them is represented twice, and strategy n is not represented at all.*

Proof: v_0 is a democracy. In the description of the algorithm, we justified that v_1 is either a democracy or satisfies property Π. In the description of the algorithm we also justified that if v_{t-1} satisfied property Π then v_t is either a democracy or it satisfies property Π. $\square$

Consider now all the vertices of the polytope that satisfy property Π, as well as all the vertices of the polytope that are democracies. We define an auxiliary graph G on these vertices, where the vertices satisfying property Π have exactly two neighbors in G, and those that are democracies have exactly one neighbor in G. In particular,

- The two neighbors (in G) of a vertex v satisfying property Π are obtained from the polytope as follows: If j is the strategy that is represented twice at v, consider un-tightening either $z_j \geq 0$ or $R_j z \leq 1$. Either one will define an edge of the polytope adjacent to v whose other endpoint is either a democracy or a vertex satisfying property Π. Set those two vertices to be the neighbors of v in G.

- The single neighbor (in G) of a vertex v that is a democracy is obtained from the polytope as follows: Since v is a democracy, either $z_n \geq 0$ or $R_n z \leq 1$ is tight. Consider un-tightening whichever inequality is tight. This defines an edge of the polytope whose other endpoint is either a democracy or a vertex satisfying property Π. Set that vertex to be the neighbor of v in G.

Clearly G comprises paths and cycles, as every vertex has degree either 1 or 2. Moreover, all democracies are endpoints of paths in G, since they have degree 1. Let's call "main path" the path whose endpoint is v_0 and whose other endpoint is some democracy $v^* \neq v_0$. We can show the following by induction.

Claim 4. *The Lemke-Howson algorithm traverses the main path starting from v_0, and stopping at v^*.*

Given the claim, the proof is concluded. $\square$

Some final remarks about the Lemke-Howson algorithm are in place.

- The algorithm provides an alternative proof that a Nash equilibrium exists in two-player games. In particular, the existence of a Nash equilibrium is implied by the correctness of the algorithm.
- Moreover, it shows that there always exists a rational equilibrium in two-player games.
- Its worst-case running time is exponential in the number of strategies. This lower bound was established by Savani and von Stengel [5].
- There are analogues of the Lemke-Howson algorithm for multi-player games working with manifolds instead of polytopes. (See Rosenmüller [4] and Wilson [6].)

References

[1] David Gale, Harold W. Kuhn and Albert W. Tucker. On Symmetric Games. *Contributions to the Theory of Games,* 1:81-87, 1950.

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

[3] John F. Nash. Noncooperative Games. *Annals of Mathematics*, 54:289–295, 1951.

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

[5] Rahul Savani and Bernhard von Stengel. Exponentially Many Steps for Finding a Nash Equilibrium in a Bimatrix Game. *In Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.

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