MIT 6.7980 — Topics in Multiagent Learning Fall 2026

Supplementary reading S2

Centralized algorithms for Nash equilibrium computation

Instructor: Prof. Constantinos Daskalakis

In previous lectures, we saw the basic game theory formalism, and some of the most fundamental equilibrium concepts, and their existence proofs. Nash's proof that a Nash equilibrium in randomized strategies exists in every finite game makes use of Brouwer's fixed point theorem, which does not immediately suggest an algorithm for computing Nash equilibria. On the other hand, we saw that the existence of Nash equilibrium in two-player zero-sum games can also be established using strong linear programming duality, which suggests a polynomial-time algorithm for computing Nash equilibria in these games.

Similarly, correlated and coarse correlated equilibria in general-sum games can also be computed in time polynomial in the game description using linear programming, as the equilibrium constraints can be written as a system of linear inequalities in the joint distribution over actions. Moreover, linear programming methods can be leveraged to obtain polynomial-time algorithms for certain families of what are called "succinct games," wherein the payoffs are sparse or have other structure that makes an explicit representation of a joint distribution over actions super-polynomial in size compared to the game's natural description. Still a correlated or coarse correlated equilibrium can be computed efficiently in many cases, using linear programming approaches such as Ellipsoid Against Hope [PR08].

In this lecture, we revisit Nash equilibrium computation in general games. We will discuss several algorithms for computing Nash equilibria. Roughly speaking those algorithms fall into two buckets. One bucket contains algorithms that directly target the equilibrium constraints, using linear programming, and more generally algorithms for solving systems of polynomial equations and inequalities. The other bucket contains algorithms that make tighter use of the fixed point nature of Nash equilibrium, and the directed parity argument underlying its existence proofs. In all cases, our algorithms will have super-polynomial complexity, unless the game has special structure. In future lectures, we will offer complexity theoretic justification is a deep reason why polynomial-time

S2.1 Support Enumeration Algorithms

To develop support enumeration algorithms, we will study whether knowing the *support* of a Nash equilibrium, i.e. the actions that are assigned non-zero probability, can reduce the computational complexity of solving for a Nash equilibrium. We will start with two-player games and proceed to general-sum games.

S2.1.1 Two-player games

Suppose we have a two-player game where one player has m actions and the other player has n actions. As described in earlier lectures, such games are commonly represented by a pair of $m \times n$ matrices (R, C). The actions of one player, called "Row," are in one-to-one correspondence with the integers $\{1, ..., m\}$ which index the rows of these matrices, and the actions of the other player, called "Column," are in one-to-one correspondence with the integers $\{1, ..., n\}$ which index the columns of these matrices. In particular, when Row plays action i and Column plays action j, they receive payoffs R_{ij} and C_{ij} respectively. When Row uses a distribution x over $\{1, ..., m\}$ and Column uses a distribution y over $\{1, ..., n\}$, they receive expected payoffs $x^T R y$ and $x^T C y$ respectively. We will assume that each payoff entry in R and C is a rational number whose numerator and denominator can be described using L bits.

Now, suppose that someone told us the supports S_R and S_C of the Row and Column players' mixed strategies, respectively, in some Nash equilibrium of the game. Using this information, we can construct the following linear program to find a Nash equilibrium (x, y):

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

The feasibility of this linear program follows from the fact that S_R and S_C are the supports in some Nash equilibrium of the game. This Nash equilibrium is a feasible solution to this linear program. In the other direction, any feasible solution to the above linear program is a Nash equilibrium. This is because if (x, y) is a feasible solution to the above linear program, then x places positive probability only on a subset of S_R and y places positive probability only on a subset of S_C. At the same time, the first couple of constraints imply that any action in S_R must be a best response to y and any action in S_C must be a best response to y. Putting these together we have the implications, which mean that (x, y) is a Nash equilibrium:

$$\begin{aligned} &\forall i : x_i > 0 \Rightarrow i \in S_R \Rightarrow i \text{ is a best response to } y; \text{ and} \\ &\forall j : y_j > 0 \Rightarrow j \in S_C \Rightarrow j \text{ is a best response to } x. \end{aligned}$$

If we don't know the supports of some Nash equilibrium, we can enumerate over all possible pairs of supports $(S_R, S_C) \subseteq [m] \times [n]$, and try to find a feasible solution of the corresponding

linear program. As a Nash equilibrium always exists, at least one of these linear programs will be feasible. So the overall running time will be $2^{m+n} \cdot \text{poly}(|\ R\ |, |\ C\ |)$, where the 2^{m+n} factor is due to trying all possible pairs of supports, and the polynomial factor in the descriptions of the matrices R and C is determined by the complexity of solving a linear program.

As a corollary of the correctness of the above algorithm, we also get a proof of the existence of Nash equilibria that use rational numbers of polynomial bit complexity in the size of the game.

Corollary S2.1. In any two-player game, there exists a Nash equilibrium whose mixed strategies use only rational numbers in their probability distributions. Moreover, these numbers have polynomial bit complexity in the bit complexity required to represent the payoff matrices of the game.

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 and any feasible solution is a Nash equilibrium. In particular, any vertex is a Nash equilibrium, and any vertex is a vector of rational numbers whose bit complexity is polynomial in the description of the LP, and hence the description of the game. □

As remarked in Lecture 3, however, the corollary is not true for k-player games where $k > 2$. Indeed, Nash's 1951 paper [Nas51] already gave an example of a 3-player game that only has irrational equilibria.

S2.1.2 n-player games

Now, let's consider how to generalize the approach to n-player games, for $n > 2$. Suppose that someone told us the support $S_i \subseteq A_i$ of each player i's mixed strategy in some Nash equilibrium of the game. Given this information, we could solve the following program to find a Nash equilibrium $x = (x_1, ..., x_n) \in \Delta(A_1) \times ... \times \Delta(A_n)$:

$$\begin{aligned} \forall \text{ player } i: \quad & u_i(a_i; x_{-i}) \geq u_i(a_i'; x_{-i}), \forall a_i \in S_i, \forall a_i' \in A_i; \\ & \sum_{a_i \in A_i} x_i(a_i) = 1; \\ & x_i(a_i) \geq 0, \forall a_i \in S_i; \\ & x_i(a_i) = 0, \forall a_i \in A_i \setminus S_i. \end{aligned}$$

Indeed, if there is a Nash equilibrium $x = (x_1, ..., x_n)$ where each x_i has support S_i, then this Nash equilibrium is a solution to the above system of polynomial equations and inequalities. In the other direction, any feasible solution $x = (x_1, ..., x_n)$ to the above system is a Nash equilibrium. Indeed, any feasible solution satisfies that for all players i, x_i assigns positive probability to a subset of S_i. Moreover, any action in S_i is a best response to x_{-i}. Putting these together we have the following implications, which mean that x is a Nash equilibrium:

$$\forall \text{ players } i, \forall a_i \in A_i: \ x_i(a_i) > 0 \Rightarrow a_i \in S_i \Rightarrow a_i \text{ is a best response to } \ x_{-i}.$$

However, notice that now $u_i(a_i; x_{-i})$ is not linear in x, but a polynomial of degree $n-1$. So the above problem amounts to solving a system of polynomial equations and inequalities in the variables x.

To analyze the running time, let us suppose for simplicity that every player has k actions. Then the above problem is a system of $M = O(n \cdot k^2)$ polynomial equations and inequalities, of degree $D = n-1$ in $N = n \cdot k$ variables.[1] This can be solved, to B bits of accuracy per variable, using tools from the existential theory of the reals [Ren92], in time

$$B \cdot (nk^n) \cdot L \cdot (MD)^{O(N)} = B \cdot L \cdot (nk)^{O(nk)},$$

where L is the number of bits needed to represent a single payoff entry in the game.[2] Enumerating over all possible supports incurs an additional factor of 2^{nk} so the overall running time to compute a Nash equilibrium with B bits of accuracy per entry is:

$$B \cdot L \cdot (nk)^{O(nk)}.$$

Recall that the bits required to represent a n-player game with k actions per player is $L \cdot n \cdot k^n$. So the running time of our algorithm could be exponential in the description of the game, e.g. when n stays constant and k goes to infinity. On the other hand, the running time is quasi-polynomial if the growth of k is bounded by a polynomial in n.[3]

S2.2 Algorithms for Symmetric Games

We will now discuss whether the running times of our algorithms from the previous section can be improved for the class of *symmetric* games.

Definition S2.2. A n-player game is called *symmetric* iff:

- All players have the same set of actions: $A_1 = \cdots = A_n = \{1, ..., k\}$; and
- there exists some function f of $k+1$ arguments such that every player i's utility can be written as: $u_i(a_i; a_{-i}) = f(a_i; n_1(a_{-i}), ..., n_k(a_{-i}))$, where $n_j(a_{-i})$ is the number of players choosing action j in action profile a_{-i}.

That is, all players have the same set of actions, and all players have the same utility function that depends on their own action and the number of other players choosing each action.

For example, rock-paper-scissors is a two-player symmetric game. Guess-$\frac{2}{3}$-of-the-average,[4] where n players submit numbers in $\{0, ..., 100\}$ and whoever is closest to 2/3s of the average

[1] We can reduce the number of constraints to $M = O(n \cdot k)$ as, if we are a bit less wasteful, we can write $O(k)$ as opposed to $O(k^2)$ constraints per player. But this won't affect the running-time asymptotics.

[2] The factor of (nk^n) on the left hand side of the afore-stated running time is for converting all the payoffs in the game, which are rational numbers with L bits in the numerator and the denominator, to integers by multiplying all numbers with their least common multiple.

[3] 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*.

[4] https://en.wikipedia.org/wiki/Guess_2/3_of_the_average

wins \$1, which is split uniformly if there are ties, is a multi-player symmetric game. Also, congestion games, where players choose paths between the same source and destination nodes in some network and they suffer traffic depending on the number of players using each edge on their path, are symmetric games. Notice that describing symmetric games can be done much more succinctly than general games. In particular, a n-player k-action symmetric game can be described by specifying $O(\min\{kn^{k-1}, k^n\})$ numbers, which are exponentially fewer compared to the $O(nk^n)$ numbers needed to describe an abitrary game, when n is large and k is small.

S2.2.1 Symmetric equilibria: existence and computation

In rock-paper-scissors, the unique Nash equilibrium of the game is symmetric, i.e. both players use the uniform mixture over their actions. More generally, a symmetric Nash equilibrium is defined as follows.

Definition S2.3. In a symmetric game, a Nash equilibrium $x = (x_1, ..., x_n)$ is called *symmetric* if $x_1 = x_2 = ... = x_n$, i.e. all players use the same mixed strategy.

While rock-paper-scissors has a symmetric Nash equilibrium, it is a priori not clear whether symmetric games ought to have a symmetric Nash equilibrium. As it turns out, this must be the case, as was shown by Nash in his 1951 paper. Indeed, Nash showed a more general statement than the statement below.

Theorem S2.4 ([Nas51])**.** In every symmetric game, there exists a symmetric Nash equilibrium.

Proof. Recall Nash's function $f : \times_i \Delta(A_i) \to \times_i \Delta(A_i)$, which maps some x to a y defined as follows, for all players i and actions $a_i \in A_i$:

$$y_i(a_i) = \frac{x_i(a_i) + \max(0, u_i(a_i; x_{-i}) - u_i(x))}{1 + \sum_{a_i' \in A_i} \max(0, u_i(a_i'; x_{-i}) - u_i(x))}.$$

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

$$\times_i \Delta(A_i) \cap \{x_1 = x_2 = ... = x_n\}.$$

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 1. And since it belongs to the restricted set, it must be a symmetric one. □

A symmetric Nash equilibrium $x = (x_1, ..., x_n)$, where $x_1 = ... = x_n$, of a n-player k-action symmetric game can be found as follows:

- Guess the support of x_i: 2^k possibilities;

- Write down a system of polynomial equations and inequalities corresponding to the Nash equilibrium conditions for the guessed support. This is a simplified version of the system we wrote down for general games in Section S2.1.2. Done well, the total number of constraints is $M = O(k)$. The polynomials involved have degree $D = n - 1$ in $N = k$ variables (c.f. $k \cdot n$ variables for general games), so the system can be solved to B bits of accuracy per variable using the existential theory of the reals in a number of operations equal to:

$$\min\{kn^{k-1}, k^n\} \cdot L \cdot (D \cdot M)^{O(N)} \equiv \min\{kn^{k-1}, k^n\} \cdot L \cdot (kn)^{O(k)},$$

  where L is the number of bits needed to represent a single payoff entry in the game.[5]

So the overall running time is $\min\{kn^{k-1}, k^n\} \cdot L \cdot (kn)^{O(k)}$, which is polynomial in the description of the game if $k = O(n)$.

S2.2.2 Symmetrization

In the previous section, we saw that, when the number of actions $k = O(n)$ and the game is symmetric, Nash equilibria can be computed in polynomial time via support enumeration and solving systems of polynomial equations and inequalities. Can we hope to get a polynomial-time algorithm when $k = \omega(n)$?

We will show that this is impossible for two-player symmetric games, unless there is a polynomial-time algorithms for arbitrary two-player games. In particular, we will show a polynomial-time reduction from the problem of computing a Nash equilibrium in general two-player games to the problem of computing a Nash equilibrium in two-player symmetric games. The reduction we present is due to Gale, Kuhn and Tucker [GKT52].

Suppose that we are given an arbitrary two-player game $\mathcal{G}_1 := (R, C)$ and we want to compute a Nash equilibrium of this game. Given the following simple exercise, we will assume, without loss of generality, that R and C have strictly positive entries, i.e. that $R, C \in \mathbb{R}_+^{m \times n}$, where m and n are, respectively, the number of actions of the row and column players.

Exercise S2.5. Show that computing a Nash equilibrium of an arbitrary game $\mathcal{G}$ can be polynomial-time reduced to the problem of computing a Nash equilibrium of a game $\mathcal{G}'$ whose payoff entries are all strictly positive.

Next, we will construct a $(m + n) \times (m + n)$ symmetric game $\mathcal{G}_2$, with the following payoff matrices in block form:

$$\mathcal{G}_2 := \begin{pmatrix} 0, 0 & R, C \\ C^T, R^T & 0, 0 \end{pmatrix}.$$

[5] As above, the factor of $\min\{kn^{k-1}, k^n\}$ in the afore-stated running time is for converting all the payoffs in the game, which are rational numbers with L bits in the numerator and the denominator, to integers by multiplying all numbers with their least common multiple.

Theorem S2.6. 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:

	x_2	y_2
x_1	$0, 0$	R, C
y_1	C^T, R^T	$0, 0$

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 S2.7. $x_1 \neq 0$ 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 her expected payoff by setting $x_1 = 0$ and adding $\|x_1\|_1$ probability to any action in the bottom block. This contradicts that $([x_1; y_1], [x_2; y_2])$ is a Nash equilibrium. □

Claim S2.8. 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 without loss of generality the row player can improve her expected payoff by switching to some $\tilde{x}_1$. Then

$$\tilde{x}_1^T R \hat{y}_2 > \hat{x}_1^T R \hat{y}_2. \tag{1}$$

Subclaim S2.9. 1 implies that the row player in $\mathcal{G}_2$ can improve her payoff by switching to $[\tilde{x}_1 \cdot \|x_1\|_1; y_1]$ from $[\hat{x}_1 \cdot \|x_1\|_1; y_1]$.

Proof Subclaim S2.9. The expected payoff of the row player in $\mathcal{G}_2$ from mixed strategy $[\tilde{x}_1 \cdot \|x_1\|_1; y_1]$ is

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

Her 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 larger, due to 1. This concludes the proof of the subclaim. □

Given Subclaim S2.9, we get a contradiction to our assumption that $([x_1; y_1], [x_2; y_2])$ is a Nash equilibrium of $\mathcal{G}_2$. □

□

We conclude that finding a Nash equilibrium in general two-player games can be polynomial-time reduced to the same problem for symmetric two-player games. An interesting open problem is whether this kind of symmetrization reduction can be generalized to games with more than 2 players.

Open Problem S2.10. Is there a polynomial-time reduction from general 3-player games to symmetric 3-player games?

S2.3 The Lemke-Howson Algorithm

Switching gears from the previous sections, we turn to algorithms for computing equilibria that exploit the fixed point nature of Nash equilibrium at a deeper level. In particular, we describe the celebrated Lemke and Howson algorithm [LH64], which was proposed in 1964 as a method to compute an exact Nash equilibrium of a two-player game (R, C) whose entries are rational numbers. This is a feasible task, as there always exists a Nash equilibrium using rational probabilities, as we have seen earlier. Indeed, the correctness proof this algorithm not only proves this fact but also that a Nash equilibrium exists. As such, the correctness proof of this algorithm provides an alternative proof of the existence of Nash equilibria in two-player games, which does not make use of Brouwer's fixed point theorem. As we will see, the proof will be reminiscent of our proof of Sperner's lemma from Lecture 2, and there is a deeper reason for that, as we will see in future lectures.

S2.3.1 Preparation: symmetry and non-degeneracy

To simplify our presentation, we will assume that the input game is a symmetric $n \times n$ game, i.e. $C = R^T$, and we will target finding a symmetric equilibrium of this game, which is guaranteed to exist by Theorem S2.4. From our work in Section S2.2.2, we can make this assumption that the game is symmetric without loss of generality. Indeed, if a given game is asymmetric we can polynomial-time reduce it to a symmetric one. The original version of the Lemke-Howson algorithm applies directly to asymmetric games, but we believe that its presentation for symmetric games is a bit simpler.

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 above 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, as long as the noise that was added to the original game is sufficiently small. This is what the following exercise asks you to do:

Exercise S2.11. 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.

S2.3.2 Main description of the algorithm

With the above setup, let us proceed to the meat of the algorithm. We make the following definition.

Definition S2.12. Action 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 actions are represented a *democracy*.

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

Lemma S2.13. 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:

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

Hence

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

and after normalization:

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

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 action," albeit this choice is arbitrary.

- **Step 0:** Start at vertex $v_0 := (0, 0, ..., 0)$.
- *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.
- Keeping all other inequalities tight, un-tighten the inequality $z_n \geq 0$ (which corresponds to our special action n). This defines an edge of the polytope adjacent to v_0.
- **Step 1:** Go to the other endpoint of this edge. If the obtained vertex v_1 is a democracy, then a Nash equilibrium has been found because $v_1 \neq 0$.
- Otherwise, one of the actions $1, ..., n-1$, say action 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).
- *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 action j_1.
- Un-tightening $z_{j_1} \geq 0$ while keeping tight all other inequalities that were tight defines an edge $(v_1 v_2) \neq (v_0 v_1)$ of the polytope.
- **Step 2:** Go to vertex v_2. If v_2 is a democracy, then stop.

  Comment: It will be shown (in the correctness analysis below) that it must be that $v_2 \neq 0$, and hence if v_2 is a democracy then $v_2/\|v_2\|_1$ is a symmetric Nash equilibrium.
- Otherwise, again some action $j_2 \neq n$ is doubly represented at v_2, all other actions in $\{1, ..., n-1\}$ are represented once, and the special action n is not represented at all.

  /*Proof: *This is because, for all actions who were singly represented at v_1, i.e. before the step was taken, their corresponding inequalities were maintained tight during the step. So they are still represented. Action 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 via the inequality that we did not un-tighten. Finally, action n was not represented before the step and since v_2 is not a democracy it is still not represented.*/
- ...
- **Step t:** At the generic step t of the algorithm, the algorithm arrives at vertex v_t and performs the following case analysis:
  - if v_t is a democracy, stop. *Comment:* It will be shown that it must be that $v_t \neq 0$.
  - if vertex v_t is not a democracy then one action j_t is represented twice, all other actions in $\{1, ..., n-1\}$ are represented once, and action n is not represented at all; the proof of this property can be done by induction on t assuming that this property holds for $v_1, ..., v_{t-1}$ and that the generic steps of the algorithm follow the description below.
  - between $R_{j_t} z \leq 1$ and $z_{j_t} \geq 0$, un-tighten the one that defines an edge $(v_t v_{t+1}) \neq (v_{t-1} v_t)$.

- for Step $t+1$, jump to v_{t+1}.

We are now ready to show that the algorithm is guaranteed to terminate at a non-zero democracy, thereby recovering a Nash equilibrium of the game.

Theorem S2.14. The Lemke-Howson algorithm will terminate and it will terminate at a non-zero democracy.

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

Claim S2.15. For all t, v_t is either a democracy, or it satisfies the following property:

Π : All of the actions in $\{1, ..., n-1\}$ are represented at v_t, exactly one of them is represented twice, and action 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. □

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 action 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 that has $v_0 = (0, ..., 0)$ as one of its endpoints and some democracy $v^* \neq v_0$ as its other endpoint. The following can be easily shown by induction.

Claim S2.16. The Lemke-Howson algorithm traverses the main path starting from v_0, moving in the direction of v^*, visiting every vertex in this path once, and terminating at v^*.

Given the claim, the proof is concluded. □

We make some final remarks about the Lemke-Howson algorithm.

- The algorithm provides an alternative proof that a Nash equilibrium exists in 2-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 2-player games.
- The proof works by virtue of a parity argument, reminiscent of the proof of Sperner's lemma. It identifies a directed path on the vertices of the polytope whose sink is a solution.
- Its worst-case running time is exponential in the number of actions. This lower bound was established by Savani and von Stengel [SV06].
- There are generalizations of the Lemke-Howson algorithm for multi-player games working with manifolds instead of polytopes. See Rosenmüller [Ros71] and Wilson [Wil71].

S2.4 Bibliography for this lecture

[PR08] C. H. Papadimitriou and T. Roughgarden, "Computing correlated equilibria in multi-player games," *Journal of the ACM (JACM)*, vol. 55, no. 3, pp. 1–29, 2008.

[Nas51] J. Nash, "Non-Cooperative Games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951, [Online]. Available: http://www.jstor.org/stable/1969529

[Ren92] J. Renegar, "On the computational complexity and geometry of the first-order theory of the reals. Part III: Quantifier elimination," *Journal of Symbolic Computation*, vol. 13, no. 3, pp. 329–352, 1992.

[GKT52] D. Gale, H. W. Kuhn, and A. W. Tucker, "On Symmetric Games," *Contributions to the Theory of Games (AM-24)*, vol. 1, pp. 81–87, 1952.

[LH64] C. E. Lemke and J. T. Howson Jr, "Equilibrium points of bimatrix games," *Journal of the Society for industrial and Applied Mathematics*, vol. 12, no. 2, pp. 413–423, 1964.

[SV06] R. Savani and B. Von Stengel, "Hard-to-solve bimatrix games," *Econometrica*, vol. 74, no. 2, pp. 397–429, 2006.

[Ros71] J. Rosenmüller, "On a generalization of the Lemke–Howson algorithm to noncooperative N-person games," *SIAM Journal on Applied Mathematics*, vol. 21, no. 1, pp. 73–79, 1971.

[Wil71] R. Wilson, "Computing equilibria of n-person games," *SIAM Journal on Applied Mathematics*, vol. 21, no. 1, pp. 80–87, 1971.