

## Computing and Approximating Nash Equilibrium

Branislav Bošanský

Artificial Intelligence Center,  
Department of Computer Science,  
Faculty of Electrical Engineering,  
Czech Technical University in Prague

*branislav.bosansky@agents.fel.cvut.cz*

April 5, 2018

# Equilibria by LCP/MILP Mathematical Programs

LCP formulation:

$$\sum_{j \in N} a_{ij} y_j + q_i = u \quad \forall i \in M$$

$$\sum_{i \in M} b_{ij} x_i + p_j = v \quad \forall j \in N$$

$$\sum_{i \in M} x_i = 1 \quad \sum_{j \in N} y_j = 1$$

$$x_i \geq 0, p_i \geq 0, y_j \geq 0, q_j \geq 0 \quad \forall i \in M, \forall j \in N$$

$$x_i \cdot p_i = 0, y_j \cdot q_j = 0 \quad \forall i \in M, \forall j \in N$$

# Equilibria by LCP/MILP Mathematical Programs

MILP formulation:

$$\sum_{j \in N} a_{ij} y_j + q_i = u \quad \forall i \in M$$

$$\sum_{i \in M} b_{ij} x_i + p_j = v \quad \forall j \in N$$

$$\sum_{i \in M} x_i = 1 \quad \sum_{j \in N} y_j = 1$$

$$w_i, z_j \in \{0, 1\}, \quad w_i \geq x_i \geq 0, \quad z_j \geq y_j \geq 0 \quad \forall i \in M, \forall j \in N$$

$$0 \leq p_i \leq (1 - w_i)Z, \quad 0 \leq q_j \leq (1 - z_j)Z \quad \forall i \in M, \forall j \in N$$

# Approximation of Nash Equilibria

We will focus on computing a Nash equilibrium in multi-player games and on approximate algorithms.

Why do we need to approximate NE in multi-player games?

	1		2	
	Left	Right	Left	Right
Top	3, 0, 2	0, 2, 0	1, 0, 0	0, 1, 0
Down	0, 1, 0	1, 0, 0	0, 3, 0	2, 0, 3

All Nash equilibria use irrational strategies even in such simple game:

$$\sigma_{row}(T) = \left(-13 + \sqrt{601}\right) / 24$$

...

# Approximation in Computer Science

What is approximation?

We want to find an approximate optimal solution efficiently (i.e., in polynomial time).

Formally, there are several schemes of approximation algorithms:

**APX** is the set of NP optimization problems that allow polynomial-time approximation algorithms with approximation ratio bounded by a constant (or constant-factor approximation algorithms for short).

# Approximation in Computer Science

- PTAS** – *polynomial time approximation scheme*: “an algorithm which takes an instance of an optimization problem and a parameter  $\varepsilon > 0$  and, in polynomial time in the instance, produces a solution that is within a factor  $(1 + \varepsilon)$  of being optimal (for min problems;  $(1 - \varepsilon)$  for max problems). The running time of a PTAS is required to be polynomial in  $n$  (size of an instance) for every fixed  $\varepsilon$  but can be different for different  $\varepsilon$ .”
- FPTAS** – *fully polynomial-time approximation scheme*: “a PTAS that requires the algorithm to be polynomial in both the problem size  $n$  and  $1/\varepsilon$ . All problems in FPTAS are fixed-parameter tractable.” Knapsack is the best example of an FPTAS problem.

# Approximation in Computer Science

*quasi-polynomial* algorithms –  $QP = \bigcup_{c \in \mathbb{N}} \text{DTIME} (2^{(\log n)^c})$

*Strongly NP-hard problems* do not admit FPTAS

# Approximation of Nash

different kinds of approximations of Nash:

- $\sigma$  is an  $\epsilon$ -approximation of Nash if

$$i \in \mathcal{N} \quad \forall \sigma'_i \in \Sigma_i \quad \sigma' := (\sigma'_i, \sigma_{-i})$$
$$\sum_{s \in \mathcal{S}} u_i(s) \sigma(s) \geq \sum_{s \in \mathcal{S}} u_i(s) \sigma'(s) - \epsilon$$

- $\sigma$  is an  $\epsilon$ -approximately well-supported NE (or  $\epsilon$ -NE), if

$$i \in \mathcal{N} \quad \forall s_i, s'_i \in \mathcal{S}_i \quad \sigma_i(s_i) > 0$$
$$\sum_{s_{-i} \in \mathcal{S}_{-i}} u_i(s_i, s_{-i}) \sigma_{-i}(s_{-i}) \geq \sum_{s_{-i} \in \mathcal{S}_{-i}} u_i(s'_i, s_{-i}) \sigma_{-i}(s_{-i}) - \epsilon$$

The above definition is an *additive approximation*.



# Approximation of Nash

We can also define a *relative approximation*:

- $\sigma$  is a relative  $\epsilon$ -approximation of Nash if

$$i \in \mathcal{N} \quad \forall \sigma'_i \in \Sigma_i \quad \sigma' := (\sigma'_i, \sigma_{-i})$$
$$\sum_{s \in \mathcal{S}} u_i(s) \sigma(s) \geq \sum_{s \in \mathcal{S}} u_i(s) \sigma'(s) - \epsilon \left| \sum_{s \in \mathcal{S}} u_i(s) \sigma'(s) \right|$$

- relative  $\epsilon$ -Nash is defined analogously

# Approximation of Nash

The simplest and the best known approximation algorithm is due to Lipton et al. [2].

## Theorem

*In a two player game, for any Nash equilibrium  $(\sigma_1^*, \sigma_2^*)$  and for any  $\epsilon > 0$  there exists, for every  $k \geq \frac{12 \ln n}{\epsilon^2}$ , a pair of  $k$ -uniform strategies  $(\sigma'_1, \sigma'_2)$ , such that:*

**1**  $(\sigma'_1, \sigma'_2)$  is a  $\epsilon$ -Nash equilibrium<sup>1</sup>

**2**  $|\sum_{s \in \mathcal{S}} u_i(s) \sigma'_i(s) - \sum_{s \in \mathcal{S}} u_i(s) \sigma_i^*(s)| < \epsilon$  for each player  $i$ , where  $\sigma'_i$  is a  $k$ -uniform strategy if it is a uniform distribution on a **multiset**  $\mathcal{S}'_i \subseteq \mathcal{S}_i$ , such that  $|\mathcal{S}'_i| = k$ .

---

<sup>1</sup>Originally, it has been stated for  $\epsilon$ -approximate Nash, but later strengthened to  $\epsilon$ -NE [4].

# Approximation of Nash

This theorem gives the possibility to devise a simple quasi-polynomial algorithm

## Corollary

*For a two-player game, there exists a quasi-polynomial algorithm for computing all  $k$ -uniform  $\epsilon$ -equilibria.*

Given an  $\epsilon > 0$ , fix  $k = \frac{12 \ln n}{\epsilon^2}$  and perform an exhaustive search. The running time is quasi-polynomial since  $k$  is bounded by  $\ln n$ .

These results generalize to multi-player games, where the number of pure strategies for each player is independent from the number of players [8].

# Approximation of Nash, negative results

## Theorem (Chen et al. [1])

*Bimatrix games do not have a fully polynomial-time approximation scheme unless every problem in PPAD is solvable in polynomial time.*

## Theorem (Daskalakis [5])

*For any constant  $\epsilon \in [0, 1)$ , it is PPAD-complete to find a relative  $\epsilon$ -Nash equilibrium in bimatrix games with payoffs in  $[-1, 1]$ . This remains true even if the expected payoffs are positive in every relative  $\epsilon$ -Nash equilibrium of the game.*

# Homotopy Methods and Computing/Approximating Equilibria [6]

A general method of using a simplification of a problem to solve the target problem.

In topology, two continuous functions from one topological space to another are called homotopic if one can be “continuously deformed” into the other.

Formally, we have two continuous functions  $f(x)$  and  $g(x)$  from a topological space  $X$  to  $Y$ . A homotopy is a continuous function  $H : [0, 1] \times X \rightarrow Y$  such that for all points  $x$  in  $X$ ,  $H(0, x) = f(x)$  and  $H(1, x) = g(x)$ .

- $g(x)$  is the target problem we want to solve
- $f(x)$  is the easy-to-solve problem
- we assume that we want to find a fixed point in both of these functions

# Homotopy Methods and Computing/Approximating Equilibria

## Theorem (Browder fixed point)

*Let  $S$  be a non-empty, compact, convex subset of  $\mathbb{R}^d$  and let  $H : [0, 1] \times S \rightarrow S$  be a continuous function. Then the set of fixed points,  $F_H = \{(\lambda, s) \in [0, 1] \times S : s = H(\lambda, s)\}$  contains a connected set,  $F_H^c$ , such that  $(\{0\} \times S) \cap F_H^c \neq \emptyset$  and  $(\{1\} \times S) \cap F_H^c \neq \emptyset$ .*

General algorithm scheme:

- formulate the target problem as a fixed-point problem
- formulate an artificial as a fixed-point problem with a starting point  $s^0$  that can be computed easily
- define  $H_{|(0,1) \times S} : (0, 1) \times S \rightarrow S$  in any way that makes  $H$  continuous on  $[0, 1] \times S$ .

# Homotopy Methods and Computing/Approximating Equilibria

In equilibrium computation:

- Homotopy function is defined on mixed strategies  
 $H : [0, 1] \times \Sigma \rightarrow \Sigma$ ,
- we seek Nash equilibrium; hence,  $g(\sigma) := \prod_{i \in \mathcal{N}} \mathcal{BR}_i(\sigma_{-i})$ ,
- for each  $t \in [0, 1)$ , the set of fixed points of  $H|_{\{t\} \times \Sigma}$  corresponds with the set of Nash equilibria of a particular perturbation of the game  $G(t)$ ,
- the starting problem is a game (with a unique Nash equilibrium) that is easily computed.

# Lemke-Howson as a Homotopy Method

Lemke-Howson algorithm as a homotopy method:

- Let  $G$  be a bimatrix game and let  $\alpha$  be a bonus sufficiently large to make any pure strategy a dominant strategy.
- Let  $G(t) := (\mathcal{N}, \mathcal{S}, \{v_i(t, \cdot)\}_{i \in \mathcal{N}})$  for  $t \in [0, 1]$ , where for player  $i$  and strategy  $s'_i$  it holds:

$$v_i(t, s) = \begin{cases} u_i(s) + (1-t)\alpha & \text{if } s_i = s'_i \\ u_i(s) & \text{otherwise} \end{cases} \quad (1)$$

- the homotopy function  $H$  is defined as

$$H(t, \sigma) = \prod_{i \in \mathcal{N}} \mathcal{BR}_i(t, \sigma_{-i}),$$

where the best responses are from  $G(t)$  and use utility function  $v_i$



# Lemke-Howson as a Homotopy Method

Recall the Lemke-Howson – the algorithm seeks a fully labeled vertex (a mixed strategy profile), where the label comes from either (1) a pure strategy is a best response, or (2) a pure strategy is played with probability 0.

Let's start in a initial game  $G(0)$  for some strategy  $s_i$ .

$s_i$  is clearly a dominating pure strategy (due to construction) and there is a unique best response of the opponent (we have nondegenerate games)  $s_{-i}$ .

We have a solution (NE) of  $G(0)$  and we can increase  $t$  towards 1. There are two possibilities:

- 1 it is also a NE in  $G(1) = G$ , or
- 2 there exist a  $0 < t' < 1$  such that another strategy is a best response (we are decreasing the impact of bonus  $\alpha$ )

# Homotopy Methods for n-Player Games

The homotopy path is piecewise linear in case of two-player games, thus can be computed exactly.

This is not true for the multi-player case – there it either needs to be approximated (e.g., using piecewise linear approximations) or a *predictor-corrector method* can be used.

Homotopy function is as before

$$H(t, \sigma) = \prod_{i \in \mathcal{N}} \mathcal{BR}_i(t, \sigma_{-i}),$$

and perturbed games  $G(t)$  have utility function  $v_i(t, \sigma) := t \cdot u_i(\sigma) + (1 - t)u_i(\sigma_i, p_{-i})$ , where  $p \in \Sigma$  is some fixed prior belief over the strategies.

# Computing a Quantal Response Equilibrium as a Homotopy Method

Recall the quantal response that is, in general, parametrized with  $\lambda$

$$\mathcal{BR}_i^{QR}(\lambda, \sigma) = \frac{\exp(\lambda \cdot u_i(s_i, \sigma_{-i}))}{\sum_{s'_i \in \mathcal{S}_i} \exp(\lambda \cdot u_i(s'_i, \sigma_{-i}))} \quad (2)$$

$\lambda = 0$  corresponds to a uniform strategy;  $\lambda = \infty$  to a Nash equilibrium.

This corresponds to a homotopy method, where  $\lambda = \frac{t}{1-t}$  for  $t \in [0, 1)$  ( $H(1, \sigma)$  is defined as before using standard best response).

# Using Polymatrix Games to solve n-Player NFGs

Polymatrix games can be used to approximate n-Player NFGs [7] using a homotopy method.

Why polymatrix approximation?

For each matrix we can use exact (and fast) Lemke-Howson algorithm.

The polymatrix game approximation has a natural interpretation: player  $i$  bilateral interaction with each other player  $j \neq i$  is approximated by averaging over the strategies of all other players  $k \neq i, j$  using their mixed strategies  $\sigma_k$  as the weights for computing the average payoff obtained from each combination of the pure strategies of players  $i$  and  $j$ .






# Recent Approximate Algorithms

The most recent complete approximate algorithm is based on dividing the simplexes (Exclusion Method for Finding Nash Equilibrium in Multiplayer Games) [9]:





- select a subsimplex of strategies of an  $n$ -player game
- run an exclusion oracle that can provably tell whether there is no Nash equilibrium (NE) in this subsimplex
- if the exclusion tells that there is no NE, we can prune this subsimplex out
- otherwise we divide this subsimplex and repeat

# References I

(besides the books)

-  X. Chen and X. Deng, “Settling the complexity of two-player nash equilibrium,” in *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 261–272, 2006.
-  R. J. Lipton, E. Markakis, and A. Mehta, “Playing Large Games using Simple Strategies,” in *Proceedings of ACM EC*, 2003.
-  C. Daskalakis, A. Fabrikant, and C. H. Papadimitriou, “The Game World Is Flat: The Complexity of Nash Equilibria in Succinct Games,” in *ICALP*, pp. 513–524, 2006.
-  C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, “The Complexity of Computing a Nash Equilibrium,” in *Proceedings of the 38th annual ACM symposium on Theory of computing*, 2006.
-  C. Daskalakis, “On the complexity of approximating a nash equilibrium,” *ACM Trans. Algorithms*, vol. 9, pp. 23:1–23:35, June 2013.

# References II

-  P. J.-J. Herings and R. Peeters, "Homotopy methods to compute equilibria in game theory," *Economic Theory*, vol. 42, no. 1, pp. 119–156, 2009.
-  S. Govindan and R. Wilson, "Computing nash equilibria by iterated polymatrix approximation," *Journal of Economic Dynamics and Control*, vol. 28, no. 7, pp. 1229 – 1241, 2004.
-  Y. Babichenko, S. Barman, and R. Peretz, "Simple approximate equilibria in large games," *EC*, pp. 753–770, 2014.
-  K. Berg and T. Sandholm, "Exclusion Method for Finding Nash Equilibrium in Multiplayer Games," *Proceedings of AAAI Conference on Artificial Intelligence*, 2017.