Exclusion Method for Finding Nash Equilibrium in Multiplayer Games

Kimmo Berg
Department of Mathematics and Systems Analysis
Aalto University, Finland

(joint with Tuomas Sandholm)
(Carnegie Mellon University)

July 28, 2016

Outline of the presentation

- Introduction and motivation
- Earlier literature
  - Classification of methods
  - Computational complexity
- Exclusion method
  - Exclusion oracle tells if Nash equilibrium is NOT in the region
  - Subdivision scheme and region selection important
- Numerical results

Introduction

	L	R
T	$1, 1, 1$	$0, 0, 0$
B	$0, 0, 0$	$0, 0, 0$

C

	L	R
T	$0, 0, 0$	$0, 0, 0$
B	$0, 0, 0$	$1, 1, 1$

D

- Normal-form game with n players and m actions
- Nash equilibrium p^*: no player can gain by deviating
- ϵ-equilibrium: $u_i(a, p^*_{-i}) \leq u_i(p^*) + \epsilon$, $\forall i, a \in A_i$
- How do you compute an (approximative) equilibrium?

Introduction (2)

- Two-player vs. multiplayer games
- Multiplayer games – nonlinear polynomial equations
- Correlated equilibrium? Zero-sum game?
- Find one vs. all equilibria
- Root of regret $r(p) = 0$; piecewise differentiable polynomial
- Regret of action a: $r_i(a, p) = u_i(a, p_{-i}) - u_i(p)$
  Regret of player i: $r_i(p) = \max_{a \in A_i} r_i(a, p)$
  Regret in the game: $r(p) = \max_i r_i(p)$

Classification of methods

- **Homotopy (path-following) methods**: trace equilibrium from easy, artificial game to the original game.
  Govindan and Wilson 2003/4, Herings and Peeters 2005, Turocy 2005, Lemke and Howson 1964
- **Polynomial equation solving and support enumeration**:
  Porter et al. 2008, Lipton and Markakis 2004
- **Function minimization and optimization formulations**:
  Sandholm et al. 2005, Chatterjee 2009, Buttler and Akchurina 2013, Borycka and Juszczuk 2013
- **Simplicial subdivision methods**:
  van der Laan, Talman, van der Heyden 1970-80s
- **Uniform-strategy enumeration methods**:
  Lipton et al. 2003, Hemon et al. 2008, Babichenko et al. 2014

Gambit algorithms on GAMUT games

Computation times (sec) and instances not solved (percentage)

Game class	gnm	ipa	enumpoly	simpdiv	liap	logit
Bertrand oligopoly	0.05 (30)	0.05 (75)	0.04 (50)	0.04 (0.4)	0.24 (99)	0.06
Bidirectional LEG	0.09 (0.3)	0.05 (58)	0.84 (1)	0.04 (2)	0.24 (99)	0.06 (0.1)
Collaboration	0.24 (0.1)	0.04	3.3 (50)	0.05	0.34 (99)	0.06 (0.3)
Congestion	0.05 (0.2)	0.05 (85)	0.05 (0.6)	0.04 (0.7)	0.21 (100)	0.05
Coordination	0.24 (2)	0.05	27 (8)	0.04	0.37 (99)	0.05 (0.3)
Covariant r=0.9	0.19 (1)	0.06 (87)	39	0.04 (20)	0.31 (99)	0.06 (0.3)
Covariant r=-0.5	0.13 (3)	0.05 (94)	36	0.04 (20)	0.31 (100)	0.05 (1)
Dispersion	1.18 (1)	0.04	10	0.04	0.44 (93)	0.05
Majority voting	0.77 (25)	0.05	0.32	0.04	0.24 (100)	0.06 (1)
Minimum effort	0.06	0.04	1.7	0.04	0.26 (98)	0.05 (0.1)
N player chicken (*)	0.05 (0.2)	0.04 (52)	0.04	0.04	0.07 (67)	0.05 (0.2)
N player PD (*)	0.04	0.04 (99)	0.04	0.04	0.05 (22)	0.04
Polymatrix	0.06 (1)	0.04 (79)	0.04 (50)	0.06 (0.4)	0.3 (92)	0.05 (0.4)
Random compound (*)	0.04	0.04 (37)	0.05	0.04	0.08 (56)	0.05
Random LEG	0.05 (1)	0.04 (59)	8.1 (2)	0.05 (4)	0.24 (99)	0.06
Random graphical	0.08 (3)	0.04 (96)	6.3 (6)	0.10 (17)	0.31 (99)	0.06 (0.3)
Traveler's dilemma	0.04	0.06	1.1	0.04	0.33 (98)	0.06
Uniform LEG	0.07 (0.4)	0.05 (55)	0.04 (17)	0.04 (10)	0.23 (99)	0.06

NO Gambit algorithm can solve ALL instances.

Earlier results

- Computing Nash is PPAD-complete in two-player general-sum games (Chen, Deng, Teng 2006/9)
- So is approximative equilibrium (Daskalakis 2013, Rubinstein 2016)
- Polynomial Parity Arguments on Directed graphs (Papadimitriou 1991)
- PPAD is believed to be hard
- Computing (approximative) Nash is FIXP-complete in multiplayer games (Etessami and Yannakakis 2010)

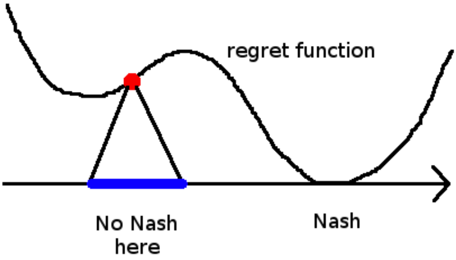
Earlier results: uniform strategies

- ϵ-equilibrium in "small" supports using k-uniform strategies
- k-uniform: probabilities are all with denominator k
- Babichenko et al. 2014: $k = O((\log m + \log n - \log \epsilon)/\epsilon^2)$
- Number of profiles: m^{nk} and $(k+1)^{nm}$
- If $n = m = 3$, $\epsilon = 10^{-3}$, $k > 10^7$ and 10^{42} points
- $O(m^{\log m})$, $O((\log n)^n)$, $O(((\log 1/\epsilon)/\epsilon^2)^c)$ (best in m)
- We improve n and ϵ: $O(c^n)$, $O(1/\epsilon^c)$, c constant (best in n)

Earlier results: solving algebraic equations

- Lipton and Markakis 2004: algebraic numbers and finite representation
- Not only approximative but close to actual Nash equilibrium
- Polynomial in $\log 1/\epsilon$, n^{nm}, L (best in ϵ)
- L is maximum bit size of payoff data

Main idea behind our method: exclusion of regions



- For any point with positive regret, the solution cannot be near this point
- Based on the function being continuous and having maximum value of derivative

Exclusion oracle

- How to determine the maximum derivative (M) of piecewise polynomial?

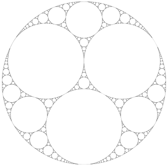
Theorem

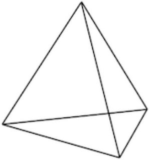
p^0-centered ball of radius s cannot contain 0-Nash if $r_i(p^0) > s \cdot M_i$ for some i

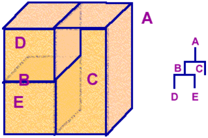
Theorem

If $r_i(p^0) \geq \epsilon$, for some $i \in N$, then region size $d < \epsilon / 2M_i$ is small enough to exclude p^0.

Subdivision scheme







- Exclude balls? Remaining regions difficult to keep track
- How to encode the regions? Simplexes?
- We use hyperrectancles (boxes)
- Easy to store min and max values in each dimension
- Split using bisection, divide along the longest edge

Region selection heuristic

- Select a region that is likely to contain Nash
- Compute ranking function based on available function values
- We use $g(R, p^0) = \max_i r_i(p^0)/(d(R) \cdot M_i(p^0))$
- R region, d its diameter, $M_i(p^0)$ maximum derivative of regret
- Favor big regions with low regret and big derivatives

Exclusion method using bisection

Repeat until ϵ-Nash found

1. Select the box with minimal value of ranking function g
2. Compute regret $r(p^0)$. If regret small enough, ϵ-Nash found. Else either exclude the box (regret is large), or bisect it along the longest edge.

Computational complexity

- $O(c^n)$, $O(c^m)$, $O(1/\epsilon^c)$
- Exponential both in n and m

Theorem

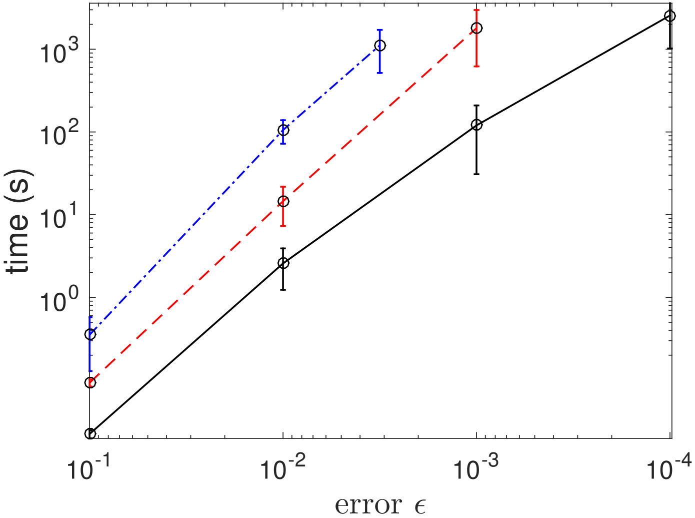
Any bisection method excludes all points with $r(p) \geq \epsilon$ within $2^{(m-1)n\lceil \log_2 \frac{2M^*}{\epsilon} \rceil}$ iterations.

Our method vs. enumeration of k-uniform profiles

Game class	Time (sec)	95% bound	Time Alg. 2	NS (%)	NS Time	NS ϵ
Bertrand oligopoly	13.7	19.3	0.01	0	-	-
Bidirectional LEG	159	337	0.013	0	-	-
Collaboration	2.8	3.7	0.0009	0	-	-
Congestion	29	71	0.027	0	-	-
Coordination	1.6	2.3	0.0009	0	-	-
Covariant r=0.9	5.5	8.4	0.006	0	-	-
Covariant r=-0.5	95	202	80	16	434	0.003
Dispersion	31	52	0.01	0	-	-
Majority voting	5.6	15.6	0.0008	0	-	-
Minimum effort	0.014	0.015	0.0008	0	-	-
N player chicken (*)	0.016	0.018	0.0008	0	-	-
N player PD (*)	0.005	0.005	0.0008	0	-	-
Polymatrix	172	358	27.2	7	373	0.003
Random compound (*)	0.014	0.015	0.001	0	-	-
Random LEG	880	1970	0.02	0	-	-
Traveler's dilemma	0.01	0.01	0.008	0	-	-
Uniform LEG	793	1850	0.02	0	-	-

Our method is the only one to solve all instances – slowly but surely.

Dependency in ϵ in random games, 3/4/5-player games



Conclusion

- Computation of equilibrium is difficult
- Fast algorithms and complete algorithms are different
- New approach for computing equilibrium
- Best upper bound in number of players n
- Development of new exclusion oracles, subdivision schemes and ranking functions
- Better bounds for derivatives of polynomials (e.g., Markov inequality 1889)
- Hybrid schemes using different methods together

Remember to live without regret...

Thank you for your attention! Any questions?