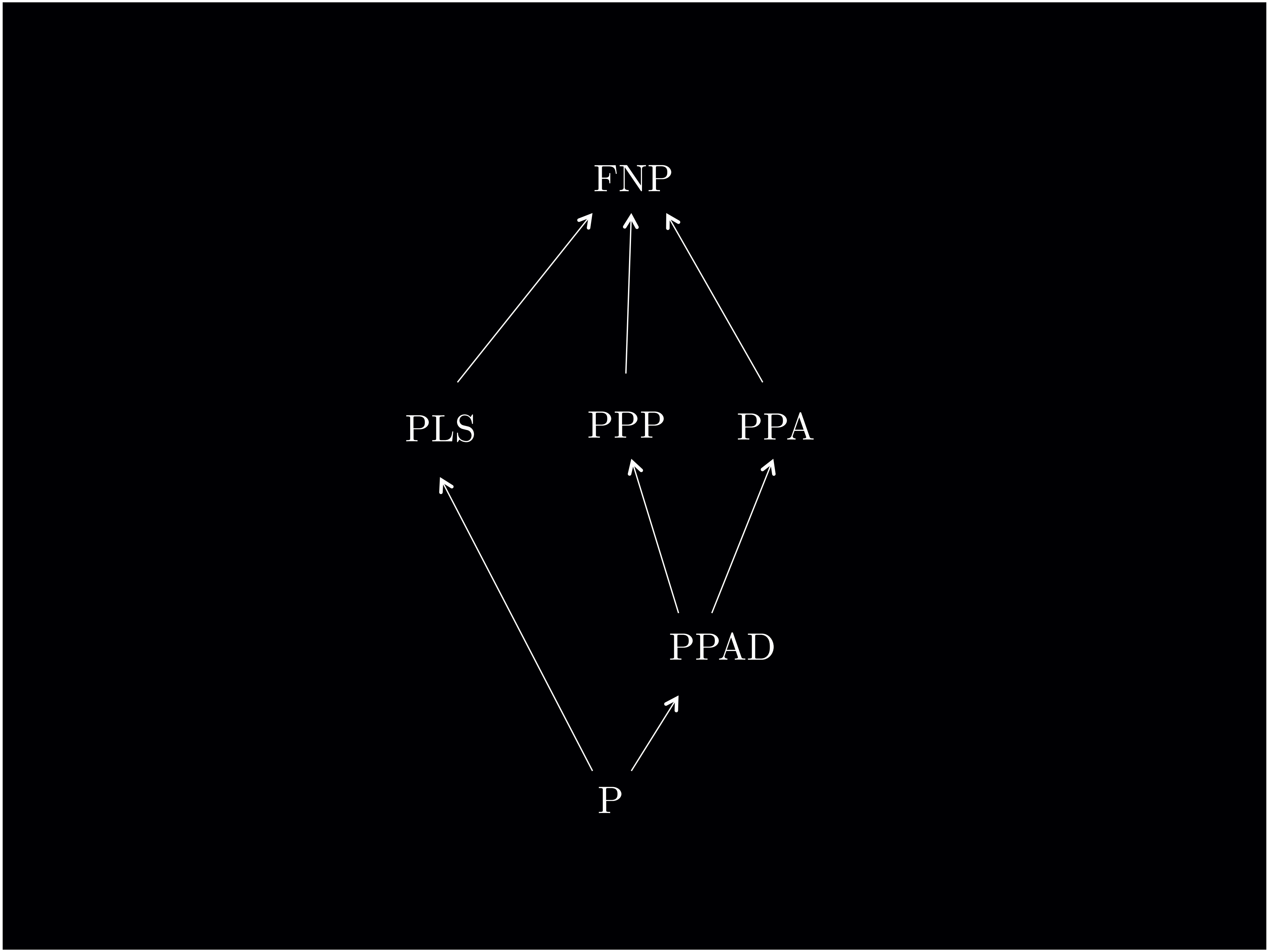
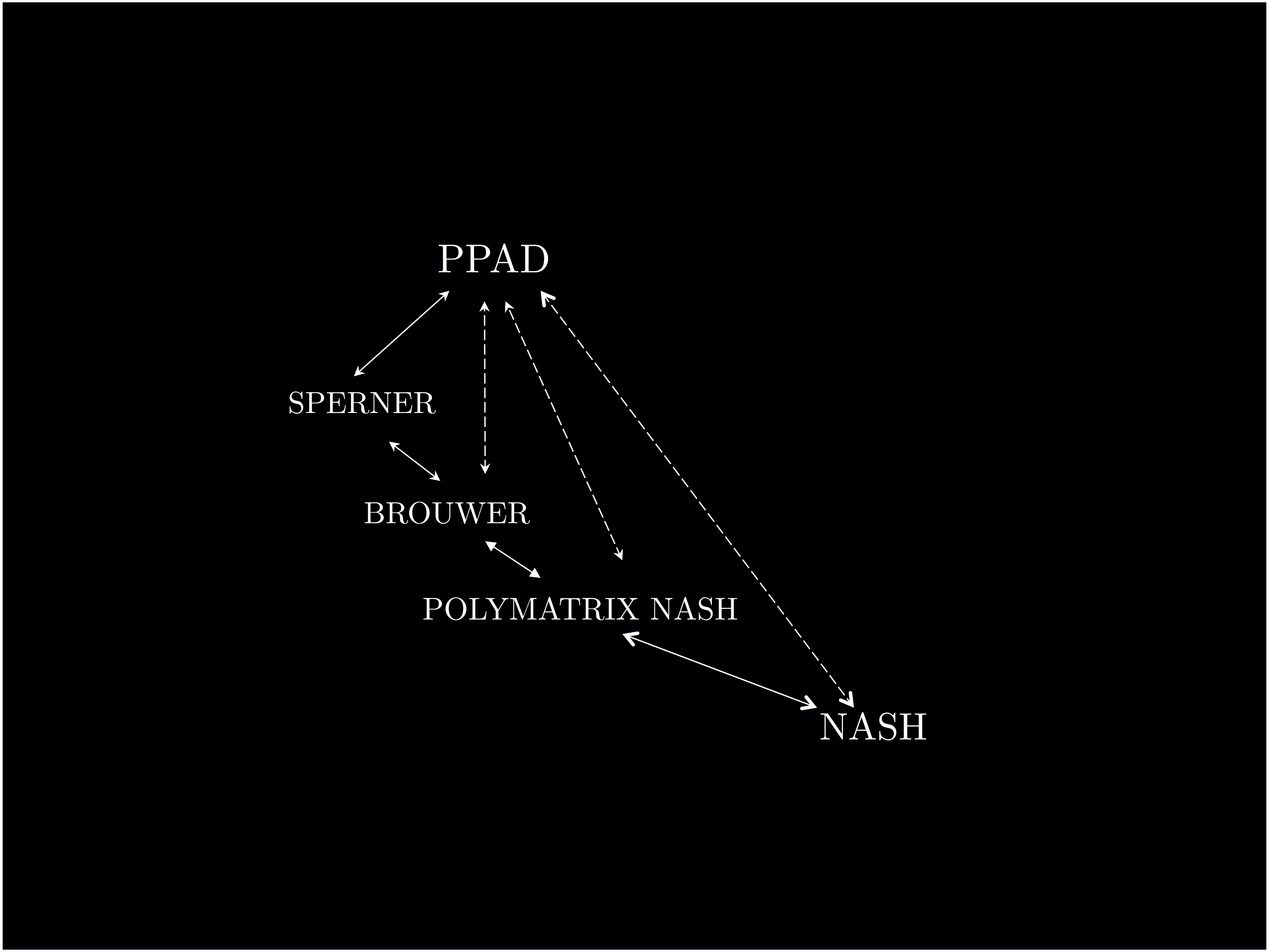
# 6.896: Topics in Algorithmic Game Theory

Lecture 11

*Constantinos Daskalakis*


FNP
PLS
PPP
PPA
PPAD
P


PPAD
SPERNER
BROUWER
POLYMATRIX NASH
NASH

# Algorithms for Nash Equilibria

- Simplicial Approximation Algorithms
- Support Enumeration Algorithms
- Lipton-Markakis-Mehta
- Algorithms for Symmetric Games
- The Lemke-Howson Algorithm

# Algorithms for Nash Equilibria

↳ Simplicial Approximation Algorithms

# Simplicial Approximation Algorithms

*suppose that S is described in some meaningful way in the input, e.g. polytope, or ellipsoid*

Given a continuous function $f : S \to S$, where *f* satisfies a Lipschitz condition and *S* is a compact convex subset of the Euclidean space, find $x \in S$ such that $|f(x) - x| < \epsilon$.

(or exhibit a pair of points violating the Lipschitz condition, or a point mapped by the function outside of *S*)

*(this is a re-iteration of the BROUWER problem that we defined in earlier lectures; for details on how to make the statement formal check previous lectures)*

Simplicial Approximation Algorithms comprise a family of algorithms computing an approximate fixed point of *f* by dividing *S* up into simplices and defining a walk that pivots from simplex to simplex of the subdivision until it settles at a simplex located in the proximity of a fixed point.

# (our own) Simplicial Approximation Algorithm

(details in Lecture 6)

1. Embed $S$ into a large enough hypercube.

2. Define an extension $f'$ of $f$ to the points in the hypercube that lie outside of $S$ in a way that, given an approximate fixed point of $f'$, an approximate fixed point of $f$ can be obtained in polynomial time.

3. Define the canonical subdivision of the hypercube (with small enough precision that depends on the Lipschitz property of $f'$ *see previous lectures* ).

4. Color the vertices of the subdivision with $n$ +1 colors, where $n$ is the dimensionality of the hypercube. The color at a point $x$ corresponds to the angle of the displacement vector $f'(x) - x$ .

5. The colors define a legal Sperner coloring.

**the non-constructive step**

6. Solve the Sperner instance, by defining a directed walk starting at the "*starting simplex*" (defined in lecture 6) and pivoting between simplices through colorful facets.

7. One of the corners of the simplex where the walk settles is an approximate fixed point.

# Algorithms for Nash Equilibria

- Simplicial Approximation Algorithms
- Support Enumeration Algorithms

# Support Enumeration Algorithms

*How better would my life be if I knew the support of the Nash equilibrium?*

… and the game is 2-player?

Setting: Let $(R, C)$ by an $m$ by $n$ game, and suppose a friend revealed to us the supports $\mathcal{S}_R$ and $\mathcal{S}_C$ respectively of the Row and Column players' mixed strategies at some equilibrium of the game.

*any feasible point (x, y) of the following linear program is an equilibrium!*

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

# Support Enumeration Algorithms

*How better would my life be if I knew the support of the Nash equilibrium?*

… and the game is 2-player?

Runtime: $2^{m+n} + \text{poly}(|R|, |C|)$

for guessing the support

for solving the LP

# Support Enumeration Algorithms

*How better would my life be if I knew the support of the Nash equilibrium?*

… and the game is polymatrix?

*input:* the support $\mathcal{S}_v$ of every node $v$ at equilibrium

*goal:* recover the Nash equilibrium with that support

➔ can do this with Linear Programming too!

the idea of why this is possible is similar to the 2-player case:

- the expected payoff of a node from a given pure strategy is linear in the mixed strategies of the other players;
- hence, once the support is known, the equilibrium conditions correspond to linear equations and inequalities.

# Rationality of Equilibria

*Important Observation:*

The correctness of the support enumeration algorithm implies that in 2-player games and in polymatrix games there always exists an equilibrium in rational numbers, and with description complexity polynomial in the description of the game!

## Algorithms for Nash Equilibria

- Simplicial Approximation Algorithms
- Support Enumeration Algorithms
- Lipton-Markakis-Mehta

# Computation of Approximate Equilibria

Theorem [Lipton, Markakis, Mehta ’03]:

For all $\epsilon > 0$ and any 2-player game with at most $n$ strategies per player and payoff entries in [0,1], there exists an $\epsilon$-approximate Nash equilibrium in which each player’s strategy is uniform on a multiset of their pure strategies of size $O\left(\frac{\log n}{\epsilon^2}\right)$.

Proof idea: (of a stronger claim)

- By Nash’s theorem, there exists a Nash equilibrium $(x, y)$.

- Suppose we take $t = \lceil 16 \log n/\epsilon^2 \rceil$ samples from $x$, viewing it as a distribution.

    $\mathcal{X}$ : uniform distribution over the sampled pure strategies

- Similarly, define $\mathcal{Y}$ by taking $t$ samples from $y$.

**Claim:** $(\mathcal{X}, \mathcal{Y})$ is an $\epsilon$-Nash equilibrium with probability at least $1 - \frac{4}{n}$.

# Computation of Approximate Equilibria

Suffices to show the following:

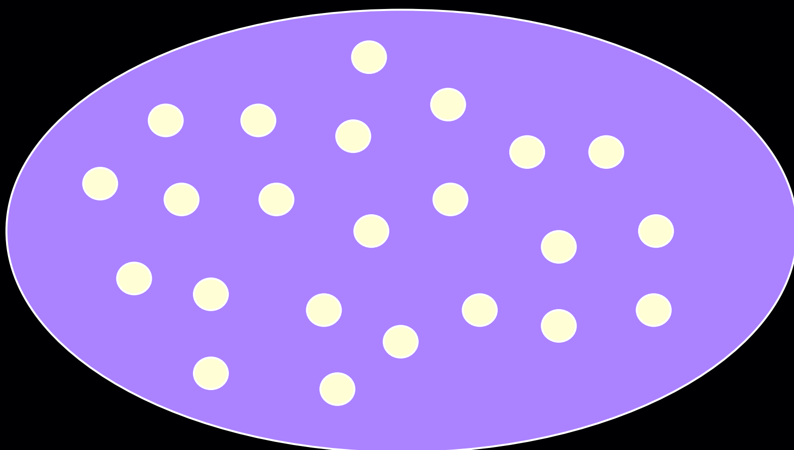
**Lemma:** With probability at least 1-4/n the following are satisfied:

$$|e_i^{\mathrm{T}} R\mathcal{Y} - e_i^{\mathrm{T}} Ry| \leq \epsilon/2, \text{ for all } i \in [n];$$

$$|\mathcal{X}^{\mathrm{T}} Ce_j - x^{\mathrm{T}} Ce_j| \leq \epsilon/2, \text{ for all } j \in [n].$$

**Proof:** on the board using Chernoff bounds.

# Computation of Approximate Equilibria



*set* $S_\epsilon$ : *every point is a pair of mixed strategies that are uniform on a multiset of size* $O\left(\frac{\log n}{\epsilon^2}\right)$.

*Random sampling from* $S_\epsilon$ *takes expected time*

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

**Oblivious Algorithm**: set $S_\epsilon$ does not depend on the game we are solving.

**Theorem** [Daskalakis-Papadimitriou '09] : *Any oblivious algorithm for general games runs in expected time* $\Omega\left(n^{(.8-34\epsilon)\log n}\right)$

## Algorithms for Nash Equilibria

- Simplicial Approximation Algorithms
- Support Enumeration Algorithms
- Lipton-Markakis-Mehta
- Algorithms for Symmetric Games

# Symmetries in Games

**Symmetric Game:** A game with $n$ players in which each player $p$ shares with the other players:

- the same set of strategies: $S = \{1, \dots, s\}$
- the same payoff function: $u = u\,(\underbrace{\sigma}_{\text{choice of } p}\,;\ \underbrace{n_1, n_2, \dots, n_s}_{\text{number of the other players choosing each strategy in } S})$

$\boxed{\textit{Description Size: } \mathrm{O}(\min\{s\, n^{s-1}, s^n\})}$

***E.g.*:**

- Rock-Paper-Scissors
- congestion games, with same source destination pairs for each player

***Nash '51*:** Always exists an equilibrium in which every player uses the same mixed strategy

# Existence of a Symmetric Equilibrium

**Recall Nash's function:**

$$f : \times_p \Delta_p \longrightarrow \times_p \Delta_p$$

$$x \overset{f}{\longmapsto} y$$

if the game is symmetric every player has the same payoff function

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

**Gedanken Experiment:**

restrict Nash's function on the set: $\times_p \Delta_p \bigcap \{x_1 = x_2 = \ldots = x_n\}$

*crucial observation:* Nash's function maps points of the above set to itself!

# Symmetrization

*[Gale-Kuhn-Tucker 1950]*

$$\begin{array}{c|c} & y \\ \hline x & R\,,\,C \end{array} \quad \Longrightarrow \quad \begin{array}{c|c|c} & x & y \\ \hline x & 0,\,0 & R,\,C \\ \hline y & C^{\mathrm{T}},\,R^{\mathrm{T}} & 0,\,0 \end{array}$$

w.l.o.g. suppose that R, C have positive entries

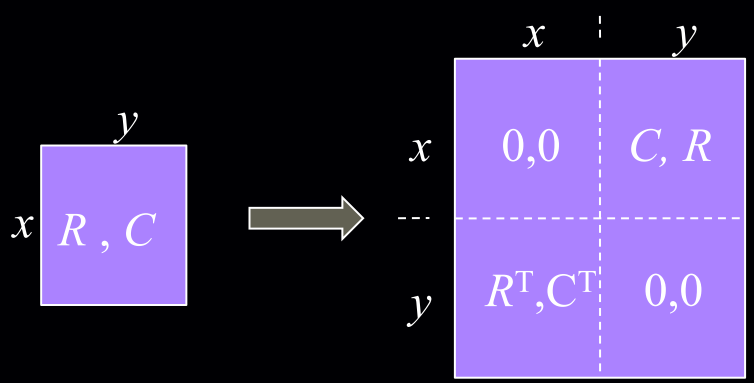
Equilibrium $\Longleftarrow$ Symmetric Equilibrium

*In fact we show that*

Equilibrium $\Longleftarrow$ Any Equilibrium

Proof: On the board.

# Symmetrization




*Hence, PPAD to solve symmetric 2-player games*

Equilibrium ⟸ Symmetric Equilibrium

*In fact* […]

Equilibrium ⟸ Any Equilibrium

***Open:*** - Reduction from 3-player games to symmetric 3-player games

- Complexity of symmetric 3-player games

# Multi-player symmetric games

If $n$ is large, $s$ is small, a symmetric equilibrium

$$x = (x_1, x_2, \ldots, x_s)$$

can be found as follows:

- guess the support of $x$ : $2^s$ possibilities

- write down a set of polynomial equations an inequalities corresponding to the equilibrium conditions, for the guessed support

- polynomial equations and inequalities of degree $n$ in $s$ variables

can be solved approximately in time

$$n^s \log(1/\varepsilon)$$

*using tools from the existential theory of the reals*

polynomial in the size of the input for $s$ up to about $\log n / \log \log n$

# *Administrativia*

Project FAQ:

*Does it have to be on computing equilibria/complexity of equilibria?*

*What would a research project vs. a survey project entail?*

*How many pages will the final write-up be?*