

# Action-Graph Games: A Compact Representation for Game Theory

Kevin Leyton-Brown

Computer Science

University of British Columbia

Based on joint papers with:

Albert Xin Jiang  
UBC

[AAAI 2006]; more recent work

Navin A.R. Bhat  
University of Toronto

[UAI 2004]

# Game Theory In One Slide ☺

- A game:
  - an interaction between two or more self-interested agents
  - each agent independently chooses an action
  - each agent derives utility from the resulting action profile
- Strategies:
  - pure strategy: picking a single action
  - mixed strategy: randomizing over actions
- Best Response:
  - I play a strategy that maximizes my own utility, given a particular (mixed) strategy profile for the other agents
- Nash Equilibrium:
  - a strategy profile with the property that every agent's strategy is a best response to the strategies of the others

# Computation-Friendly Game Representations

- **Goal:** use game theory to model real-world systems
  - allow large numbers of agents and actions
  - just consider games in **normal form**:
    - no extensive form
    - no Bayesian games
  - motivating examples in this talk will concern location games
- **Problem:** interesting games are **large**; computing equilibrium, best response, etc. is **hard**
- **Solution:**
  - compact representation
  - tractable computation

# Past Work on Compact Games

- **Temporal Structure**
  - extensive form
- **Independence**
  - some pairs of agents have no (direct) effect on each other's payoffs  
[La Mura, 2000], [Kearns, Littman, Singh, 2001], [Vickrey & Koller, 2002],  
[Oritz & Kearns, 2003], [Blum, Shelton, Koller, 2003]
  - graphical games
- **Context-Specific Independence**
  - whether agents affect each other's payoffs can depend on the action choices they each make  
[Rosenthal, 1973], [Monderer & Shapley, 1996]
  - congestion/potential games

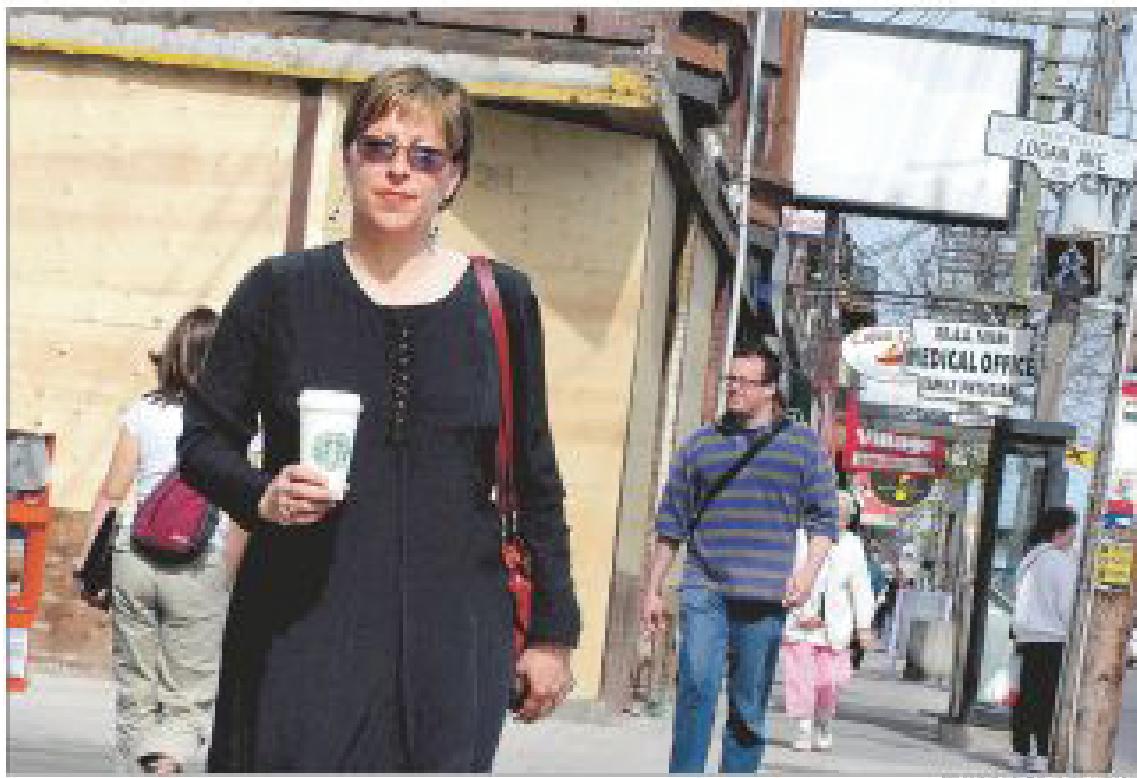
# Overview on Action-Graph Games

1. Definition and Examples
2. Analyzing the Representation
3. Computing with Games
4. Computing with AGGs
5. Experimental Results

# REPORT ON BUSINESS

MONDAY 01.05.06

Real estate agents are happy when Starbucks decides to open a new location in a neighbourhood in which they work. They say the upscale coffee chain's choice of where to locate is usually a harbinger of bidding wars to come.



Real estate agent Diane Mahon walks past the site of a soon-to-open Starbucks coffee shop in Toronto's Leslieville area.

# The Coffee Shop Problem



Web Images Groups News Local New! more »

category: Coffee Houses

e.g., "hotels in calgary" or "5000 dufferin street, toronto"

Search

Search the map  
Find businesses  
Get directions

## Local

[Print](#) [Email](#) [Link to this page](#)

Search results for **category: Coffee Houses** in this map

**A Connoisseurs' Coffee**

1075 Georgia Street West, Vancouver, BC V6E 3C9  
(604) 683-1486

**B Meliches Coffeehouse**

1244 Davie Street, Vancouver, BC V6E 1N3  
(604) 689-5282

**C Hole In The Wall Cappuccino Bar**

1030 Georgia Street West, Vancouver, BC V6E 2Y3  
(604) 646-4653

**D Starbucks Coffee Co**

1055 W Georgia, Vancouver, BC V5K 1A1  
(604) 685-5882

**E Five Roses Bakery Cafe**

1220 Bute Street, Vancouver, BC V6E 1Z8  
(604) 669-8989

**F Starbucks Coffee Co**

1095 Howe Street, Vancouver, BC V6Z 1P6  
(604) 685-7083

**G Uptown Espresso**

808 Nelson Street, Vancouver, BC V6Z 2H2  
(604) 689-1920

**H Caffe Artigiano**

763 Hornby Street, Vancouver, BC V6Z 1S2  
(604) 696-9222

**I Skyline Expresso**

900 Howe Street, Vancouver, BC V6Z 2M4  
(604) 683-4234

**J Fahrenheit Celsius Coffee**

1225 Burrard Street, Vancouver, BC V6Z 1Z5  
(604) 682-6675

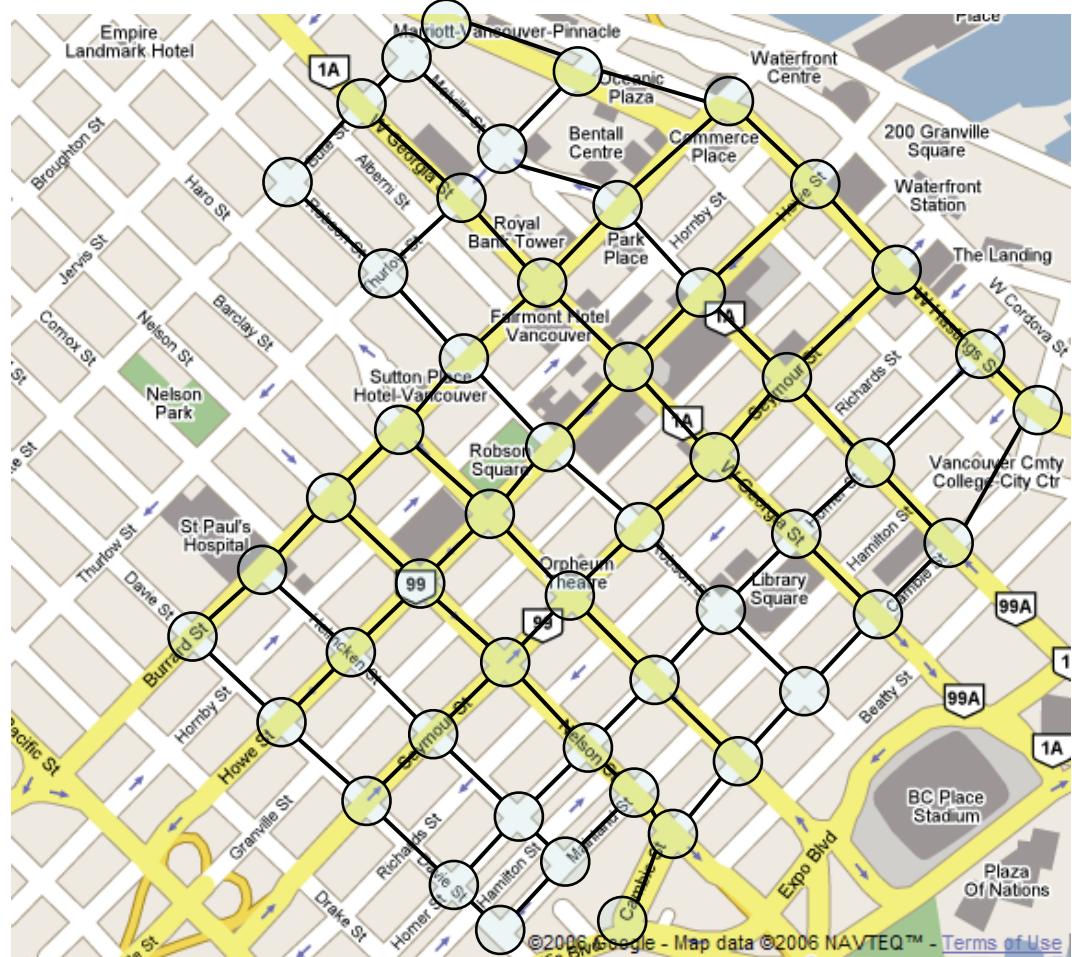
**K Chicco Dall Oriente**

1504 Robson Street, Vancouver, BC V6G 1C2



# Action-Graph Games

- set of **players**: want to open coffee shops
- **actions**: choose a location for your shop, or choose not to enter the market
- **utility**: profitability of a location
  - some locations might have more customers, and so might be better *ex ante*
  - utility also depends on the number of other players who choose the same or an adjacent location



# Formal Definitions

**Definition 1 (action graph)** An **action graph** is a tuple  $(\mathcal{A}, E)$ , where  $\mathcal{A}$  is a set of nodes corresponding to distinct actions and  $E$  is a set of directed edges.

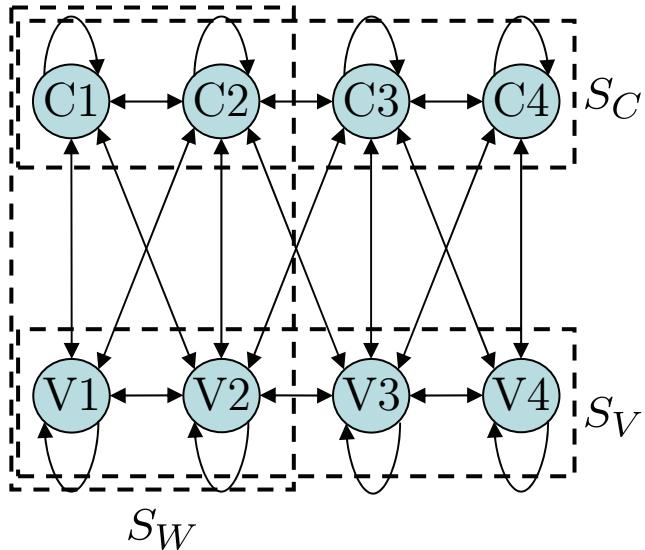
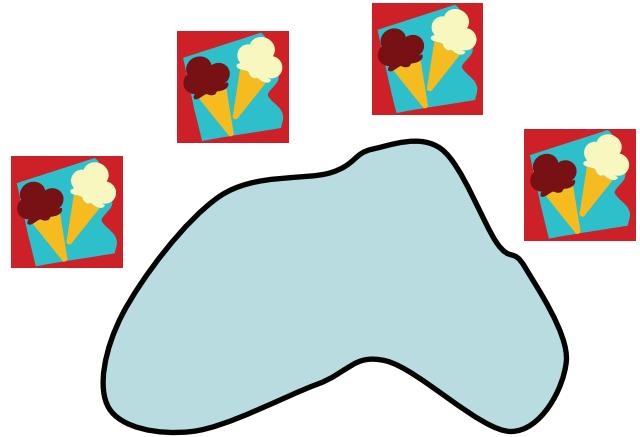
Let  $A = (A_1, \dots, A_n)$  be a **set of actions** available to each of  $n$  agents, with  $\mathcal{A} = \bigcup_{i \in N} A_i$ .

**Definition 2 (configuration)** Given an action graph  $(\mathcal{A}, E)$  and a set of action profiles  $A$ , a **configuration**  $D$  is a tuple of  $|\mathcal{A}|$  non-negative integers, where the  $j^{\text{th}}$  element  $D(j)$  is interpreted as the number of agents who chose the  $j^{\text{th}}$  action  $a_j \in \mathcal{A}$ , and where there exists some  $a \in A$  that would give rise to  $D$ . Denote the set of all configurations as  $\Delta$ .

# Elaborated Ice Cream Vendor Problem

Inspired by [Hotelling, 1929]

- vendors sell either chocolate or vanilla ice cream at one of four stations along a beach
  - chocolate (C) vendors;
  - vanilla (V) vendors;
  - can sell C/V, but only on the west side.
    - competition between nearby sellers of same type; synergy between nearby different types



Notes:

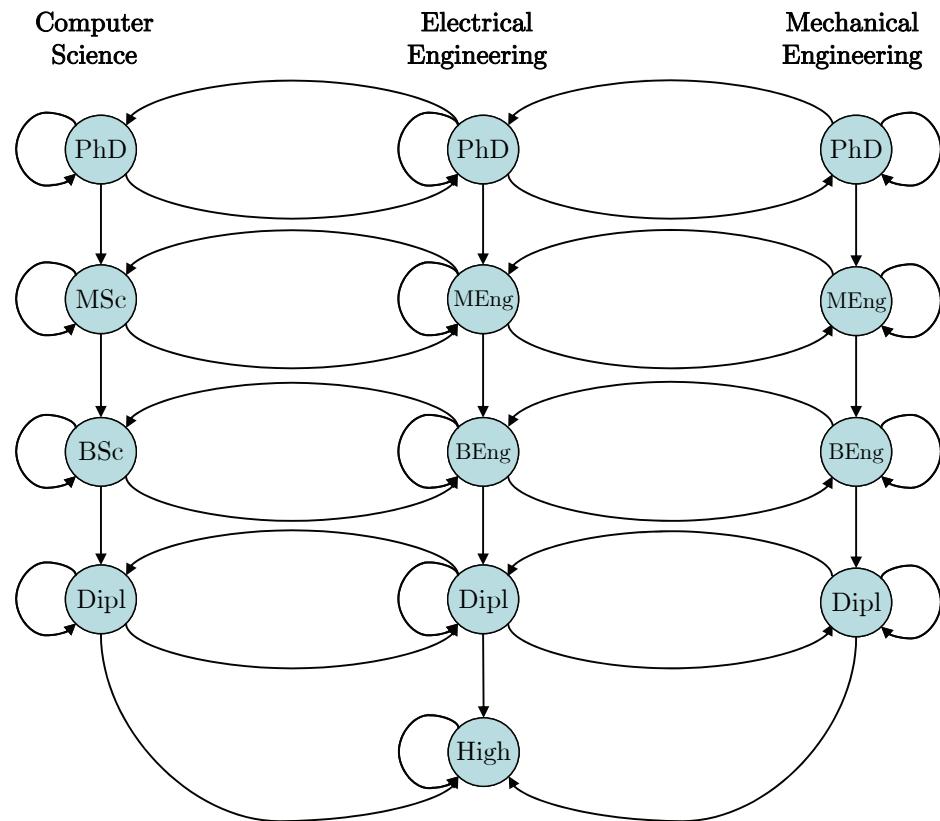
- graph structure independent of # agents
- overlapping action sets
- context-specific independence without strict independence

# The Job Market Problem

Each player chooses a level of training

Players' utilities are the sum of:

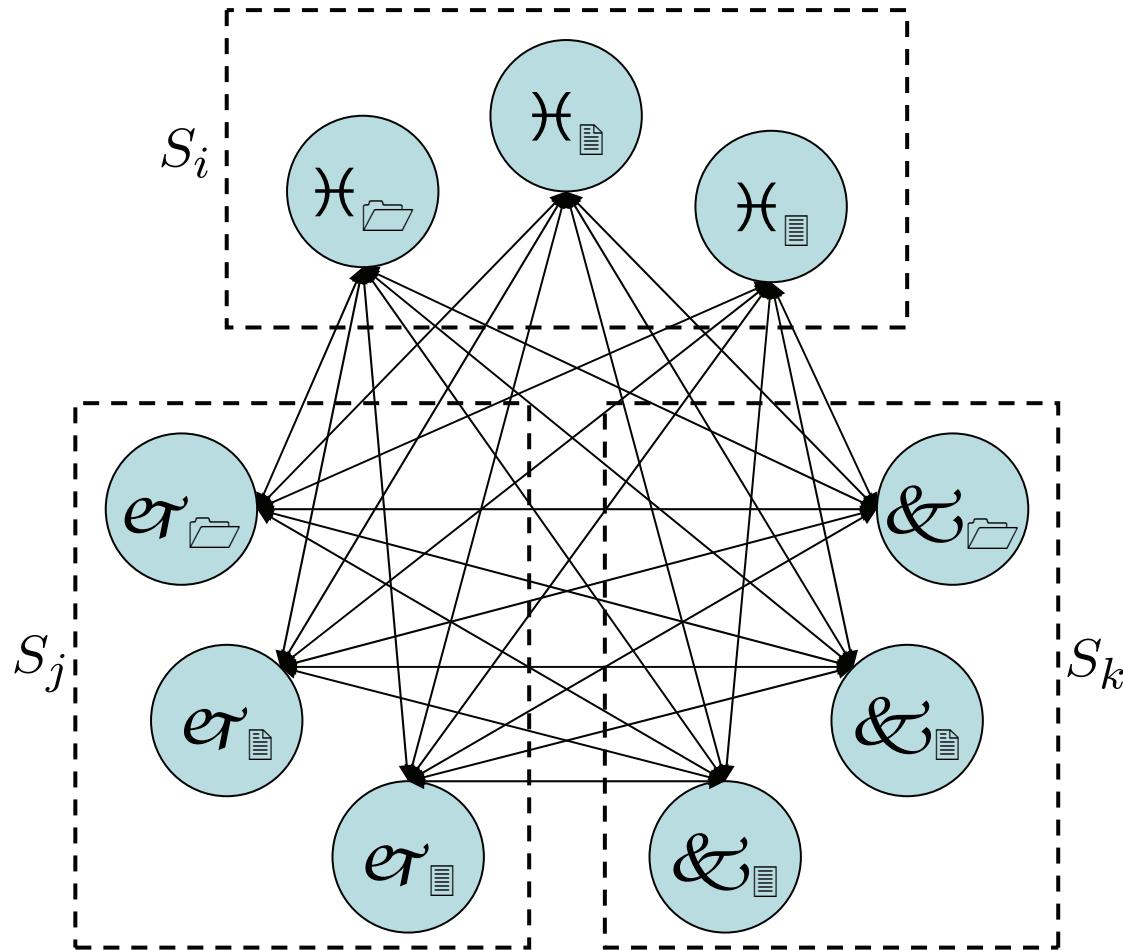
- a constant cost:
  - difficulty; tuition; foregone wages
- a variable reward, depending on:
  - How many jobs prefer workers with this training, and how desirable are the jobs?
  - How many other jobs are willing to take such workers as a second choice, and how good are these jobs?
    - Employers will take workers who are overqualified, but only by one degree.
    - They will also interchange similar degrees, but only at the same level.
  - How many other graduates want the same jobs?



# Overview on Action-Graph Games

1. Definition of AGGs and Examples
2. Analyzing and Extending the Representation
  3. Computing with Games
  4. Computing with AGGs
5. Experimental Results

# AGGs are Fully Expressive



# Analyzing the AGG Representation

AGGs are **more compact than the normal form** when the game exhibits either or both of the following properties:

## 1. Context-Specific Independence:

- pairs of agents can choose actions that are not neighbors in the action graph

## 2. Anonymity:

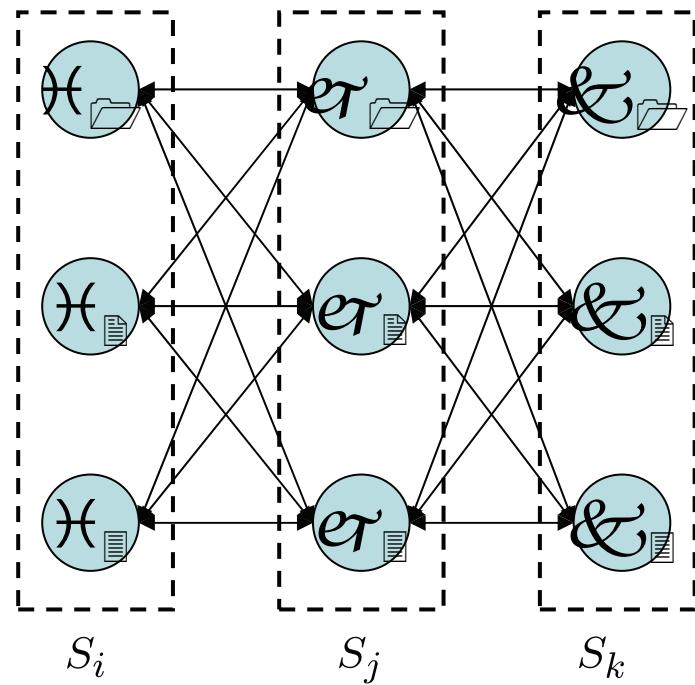
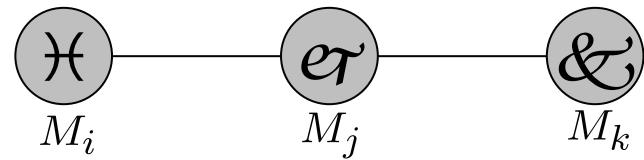
- multiple action profiles yield the same configuration

# Size of the AGG representation

How many payoffs do we need to store in an AGG?

- Bounded by  $|S| \frac{(n - 1 + \mathcal{I})!}{(n - 1)! \mathcal{I}!}$ 
  - where  $\mathcal{I}$  is the max in-degree of the action graph
- When  $\mathcal{I}$  is bounded by a constant:
  - polynomial size:  $\mathcal{O}(|S|^{\mathcal{I}})$
  - in contrast, size of normal form is  $\mathcal{O}(|S|^n)$
- Asymptotically, never larger than the normal form

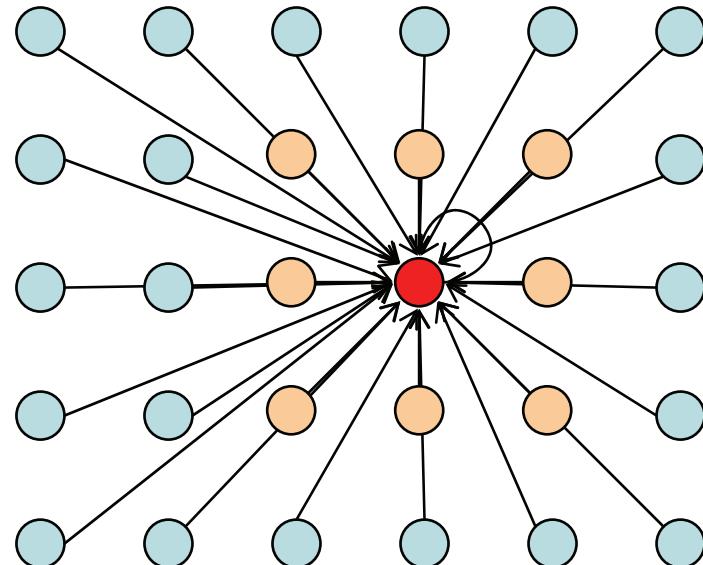
# Graphical Games are Compact as AGGs



GG	AGG
Agent node	Action set box
Edge	Bipartite graphs between action sets
Local game matrix	Node utility function

# The Coffee Shop Problem Revisited

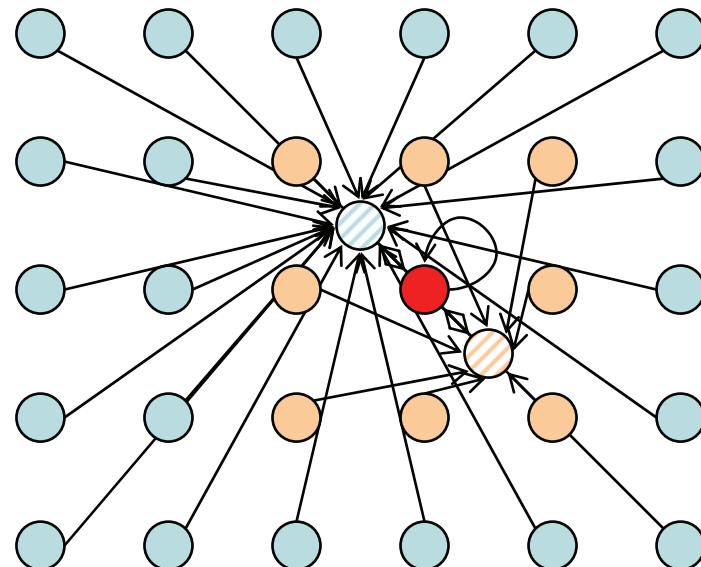
- What if utility also depends on total # shops?
- Now action graph has in-degree  $|\mathcal{A}|$ 
  - NF & Graphical Game representations:  $\beth(|\mathcal{A}|^\star)$
  - AGG representation:  $\beth(\star \ |\mathcal{A}|)$
  - when  $|\mathcal{A}|$  is held constant, the AGG representation is polynomial in  $\star$ 
    - but still doesn't effectively capture game structure
    - given  $\star$ 's action, his payoff depends only on 3 quantities!



$6 \times 5$  Coffee Shop Problem: projected action graph at the red node

# Function Nodes

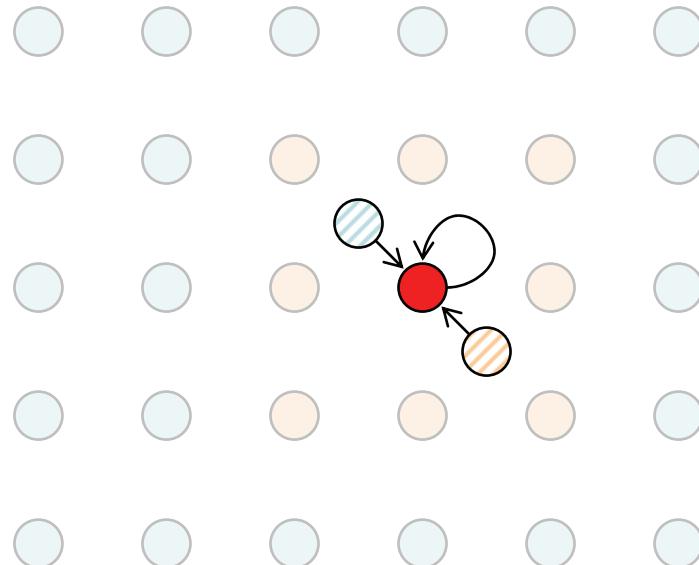
- To exploit this structure, introduce **function nodes**:
  - The “configuration” of a function node is a (given) function of the configuration of its neighbors:  $\Phi(\square) = \chi_{\square}(\Phi(v(\square)))$
- **Coffee-shop example**: for each action node  $\diamond$ , introduce:
  - One function node with adjacent actions as neighbours
    - $\Phi(\square'.) =$  total # of shops in surrounding nodes
  - Similarly, a function node with non-adjacent actions as neighbours



$6 \times 5$  Coffee Shop Problem: function nodes for the red node

# The Coffee Shop Problem

- Now the red node has only **3 incoming edges**:
  - itself, the blue function node and the orange function node
  - so, the action-graph now has in-degree 3
- Size of representation is now  $\mathbb{P}(\mathbb{X}^3)!$



$6 \times 5$  Coffee Shop Problem: projected action graph at the red node

# Overview on Action-Graph Games

1. Definition of AGGs and Examples
2. Analyzing and Extending the Representation
3. Computing with Games
4. Computing with AGGs
5. Experimental Results

# Computing with Games

Expected payoff of agent  $\mathcal{H}$  for playing action  $\cdot_{\mathcal{H}}$ ,  
if other agents play according to mixed-strategy profile  $\sigma_{-\mathcal{H}}$ :

$$V_{s_i}^i(\sigma_{-i}) \equiv \sum_{\mathbf{s}_{-i} \in \mathbf{S}_{-i}} u_i(s_i, \mathbf{s}_{-i}) Pr(\mathbf{s}_{-i} | \sigma_{-i})$$

Two **useful computations** based on  $V_{s_i}^i(\sigma_{-i})$ :

$$1. \text{ best response}(\sigma_{-i}) = \arg \max_{s_i} V_{s_i}^i(\sigma_{-i})$$

$$2. \frac{\partial V_{s_i}^i(\sigma_{-i})}{\partial \sigma_{i'}(s_{i'})} \equiv \nabla V_{s_i, s_{i'}}^{i, i'}(\sigma_{-\{i, i'\}})$$
$$= \sum_{\mathbf{s}_{-\{i, i'\}} \in \mathbf{S}_{-\{i, i'\}}} u_i(s_i, s_{i'}, \mathbf{s}_{-\{i, i'\}}) Pr(\mathbf{s}_{-\{i, i'\}} | \sigma_{-\{i, i'\}})$$

# Computing with Games

Why might we want to compute  $V_{s_i}^i(\sigma_{-i})$  or  $\nabla V_{s_i, s_{i'}}^{i, i'}(\sigma_{-\{i, i'\}})$ ?

- Best Response
- Payoff Jacobian (Govindan-Wilson Algorithm; Nash equilibrium)
- Iterated Polymatrix Approximation (IPA)
  - a quick start for the Govindan-Wilson algorithm
- Gradient for policy search multiagent RL algorithms
- Simplicial Subdivision Algorithm (Nash equilibrium)
- Papadimitriou's Algorithm (correlated Nash equilibrium)

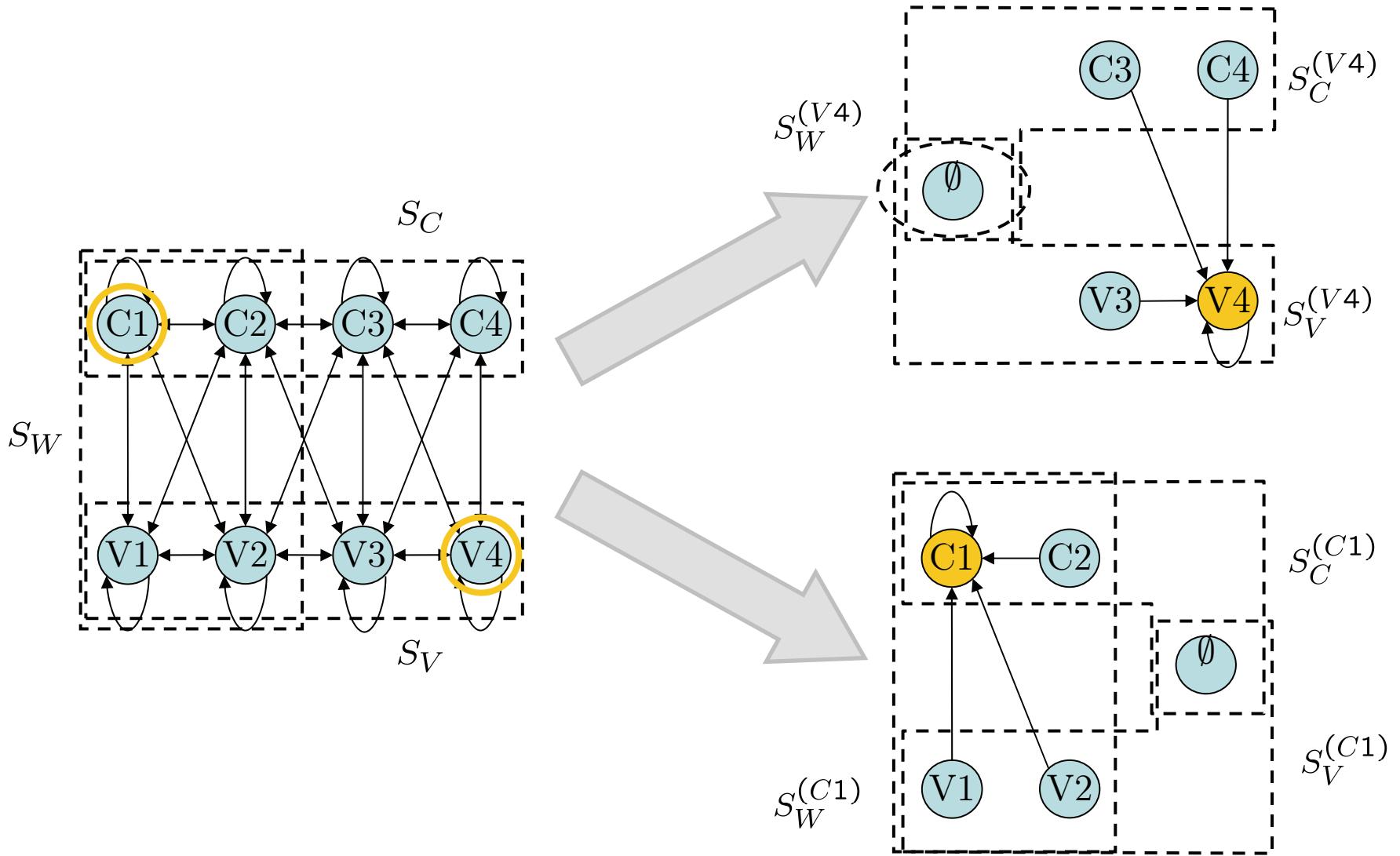
$$V_{s_i}^i(\sigma_{-i}) = \sum_{\mathbf{s}_{-i} \in \mathcal{S}_{-i}} u_i(s_i, \mathbf{s}_{-i}) Pr(\mathbf{s}_{-i} | \sigma_{-i})$$

Computational complexity:  $O(|S|^{n-1})$

# Overview on Action-Graph Games

1. Definition of AGGs and Examples
2. Analyzing and Extending the Representation
3. Computing with Games
4. Computing with AGGs
5. Experimental Results

# Computing with AGGs: Projection



# Computing with AGGs: Projection

- Projection captures context-specific independence and strict independence

$$V_{s_i}^i(\bar{\sigma}) = \sum_{\bar{\mathbf{s}}^{(s_i)} \in \bar{\mathbf{S}}^{(s_i)}} u^{s_i} \left( \mathcal{D}(s_i, \bar{\mathbf{s}}^{(s_i)}) \right) Pr \left( \bar{\mathbf{s}}^{(s_i)} | \bar{\sigma}^{(s_i)} \right)$$

$$Pr \left( \bar{\mathbf{s}}^{(s_i)} | \bar{\sigma}^{(s_i)} \right) = \prod_{j \in \bar{N}} \bar{\sigma}_j^{(s_i)}(\bar{\mathbf{s}}_j^{(s_i)}).$$

$*^{(s)}$   $\equiv$  projection with respect to action  $s$

$\bar{*}$   $\equiv$   $*_{-i}$

$\mathcal{D}(\mathbf{s})$   $\equiv$  configuration caused by  $\mathbf{s}$

# Computing with AGGs: Anonymity

- Writing in terms of the configuration captures **anonymity**

$$V_{s_i}^i(\bar{\sigma}) = \sum_{\bar{D}^{(s_i)} \in \bar{\Delta}^{(s_i)}} u^{s_i} \left( \mathcal{D} \left( s_i, \bar{D}^{(s_i)} \right) \right) Pr \left( \bar{D}^{(s_i)} | \bar{\sigma}^{(s_i)} \right)$$

$$Pr \left( \bar{D}^{(s_i)} | \bar{\sigma}^{(s_i)} \right) = \sum_{\bar{s}^{(s_i)} \in \mathcal{S} \left( \bar{D}^{(s_i)} \right)} Pr \left( \bar{s}^{(s_i)} | \bar{\sigma}^{(s_i)} \right)$$

$*^{(s)}$   $\equiv$  projection with respect to action  $s$

$\bar{*} \equiv *_{-i}$

$\mathcal{D}(\mathbf{s}, D) \equiv$  configuration caused by  $\mathbf{s}, D$

$\mathcal{S}(D) \equiv$  class of  $D$ , i.e. set of pure action profiles corresponding to  $D$

# Computing with AGGs: Anonymity

$$V_{s_i}^i(\sigma) = \sum_{\overline{D}^{(s_i)} \in \overline{\Delta}^{(s_i)}} u^{s_i} \left( \mathcal{D} \left( s_i, \overline{D}^{(s_i)} \right) \right) Pr \left( \overline{D}^{(s_i)} | \overline{\sigma}^{(s_i)} \right)$$
$$Pr \left( \overline{D}^{(s_i)} | \overline{\sigma}^{(s_i)} \right) = \sum_{\overline{s}^{(s_i)} \in \mathcal{S} \left( \overline{D}^{(s_i)} \right)} Pr \left( \overline{s}^{(s_i)} | \overline{\sigma}^{(s_i)} \right)$$

- **Good news:**
  - $\overline{\Delta}^{(s_i)}$ , the number of different configurations, is polynomial
  - thus, the first sum is over **polynomially-many** elements
- **Bad news:**
  - $\mathcal{S}(\overline{D}^{(s_i)})$ , the number of pure-action profiles corresponding to a given configuration, is exponential in the number of agents
  - thus, the second sum is over **exponentially-many** elements

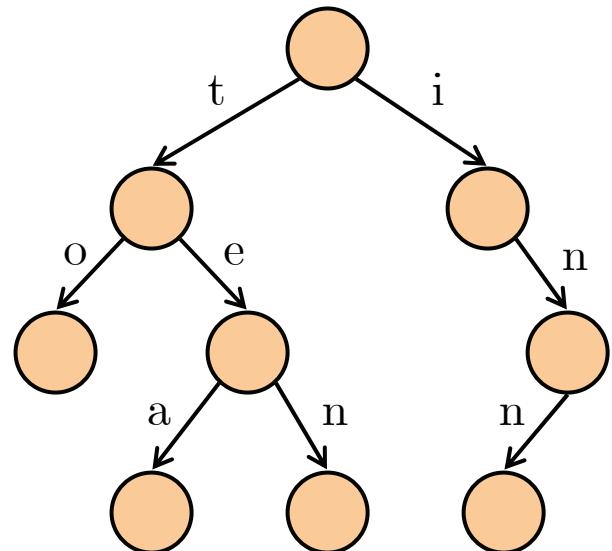
# Dynamic Programming

- A **ray of hope**: note that
  - the players' mixed strategies are independent
    - i.e.  $\sigma$  is a product probability distribution
  - each player affects the configuration  $\P$  independently
- We can use **dynamic programming** to compute the probability of a configuration:
  - base case: zero agents and the mixed strategy  $\sigma_0$ :
    - $\Delta_0 = \{\P_0\}$
    - $\P_0 = \{0, \dots, 0\}$
    - $\mathbb{P}_0(\P_0) = 1$
  - then add agents **one by one**:
    - $\Delta_{\&}$ : the set of configurations that can be built by adding any action from the support of player  $\&$ 's mixed strategy to any configuration from  $\Delta_{\&-1}$
    - $P_k(D_k) = \sum_{\substack{(D_{k-1}, s_k), \\ \mathcal{D}(D_{k-1}, s_k) = D_k}} \sigma_k(s_k) \cdot P_{k-1}(D_{k-1})$

# Dynamic Programming

- Our algorithm makes a **polynomial** number of updates
  - # **configurations** (for a given number of agents) is polynomial
  - cost of **adding an agent**: # configurations  $\times$  # actions
  - we need a data structure to manipulate probability distributions over configurations (sequences of integers) which permits quick lookup, addition and enumeration

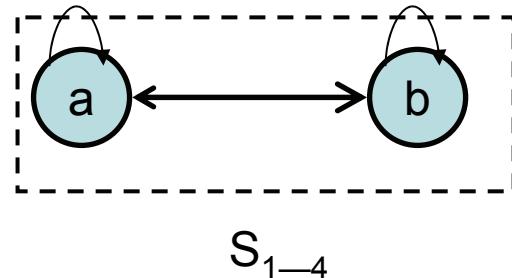
- **Tries** fit the bill
  - often used to store dictionaries (e.g., spell checker)
    - for AGGs, we store strings of integers rather than characters
  - both lookup and insertion complexity is linear (# actions)
  - enumeration can also be done in linear time (# configurations)



a trie storing 4 strings:  
to, tea, ten, inn

# AGG Computation Example

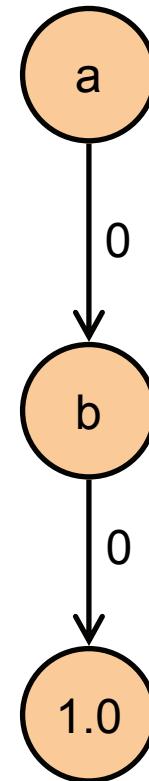
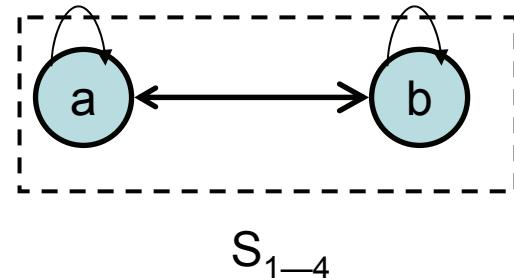
- Example game:
  - 4 players, 2 actions
- Compute joint probability distribution  $\sigma$  where
$$\sigma_1=(1, 0), \sigma_2=(0.2, 0.8),$$
$$\sigma_3=(0.4, 0.6), \sigma_4=(0.5, 0.5)$$



# AGG Example: 0 players

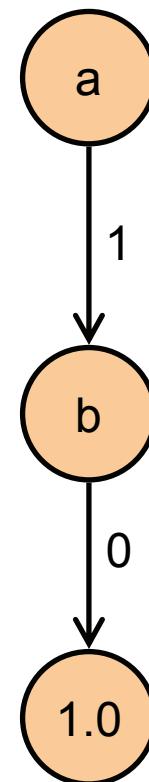
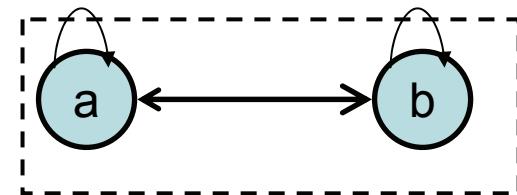
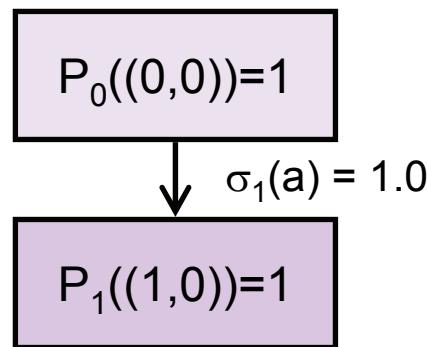
- Example game:
  - 4 players, 2 actions
- Compute joint probability distribution  $\sigma$  where
$$\sigma_1=(1, 0), \sigma_2=(0.2, 0.8),\sigma_3=(0.4, 0.6), \sigma_4=(0.5, 0.5)$$

$$P_0((0,0))=1$$



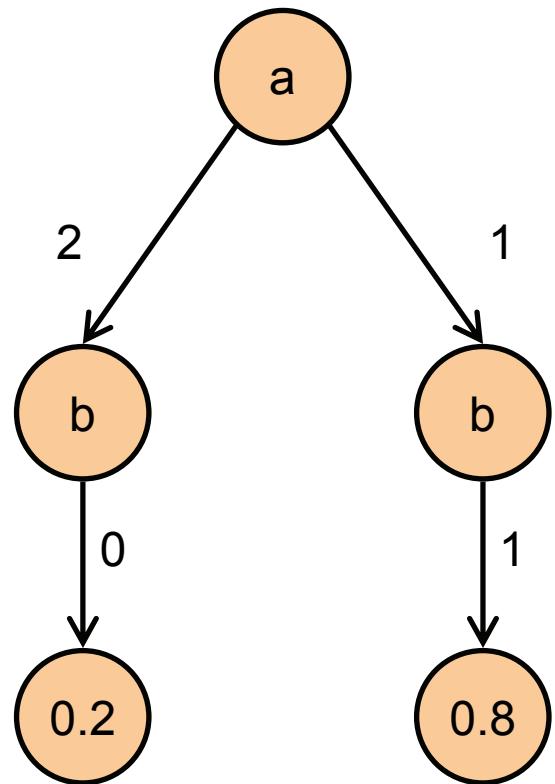
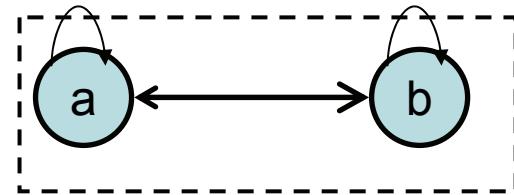
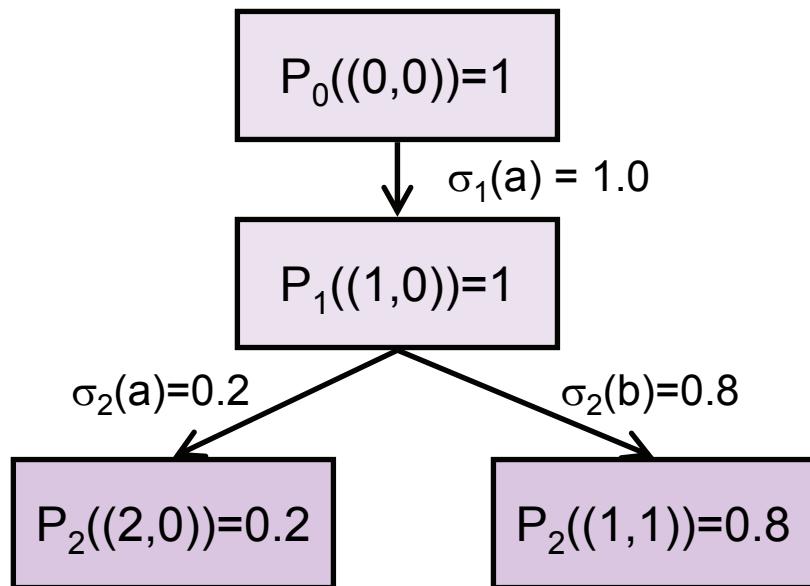
# AGG Example: 1 player

$\sigma_1=(1, 0)$ ,  $\sigma_2=(0.2, 0.8)$ ,  
 $\sigma_3=(0.4, 0.6)$ ,  $\sigma_4=(0.5, 0.5)$



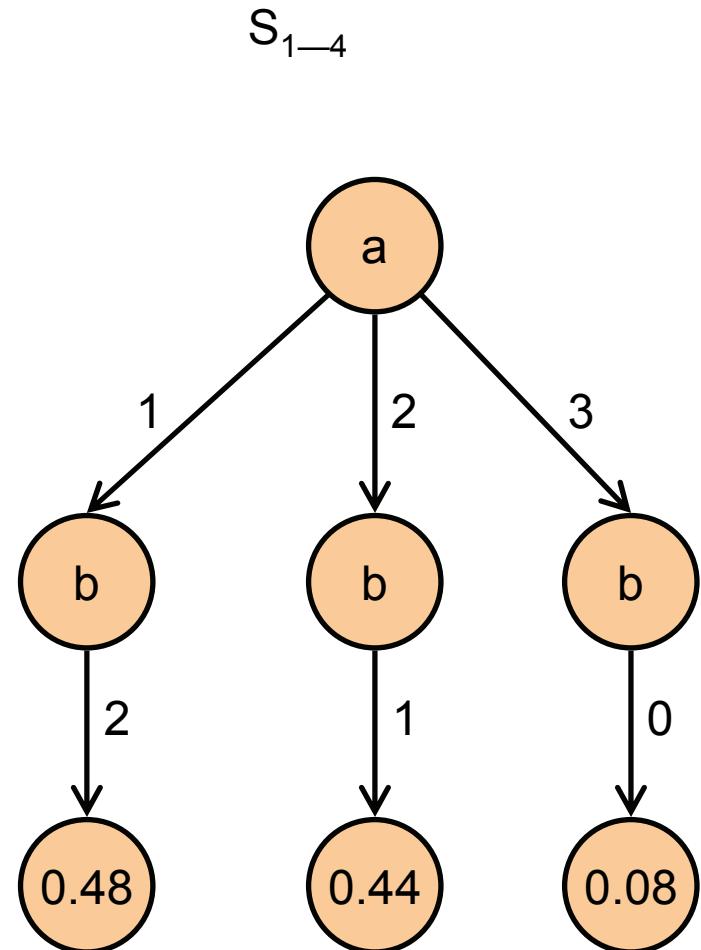
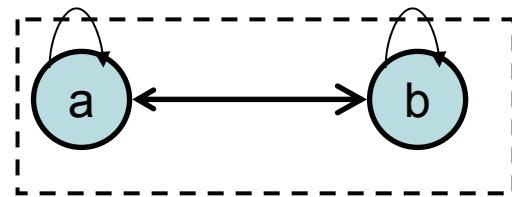
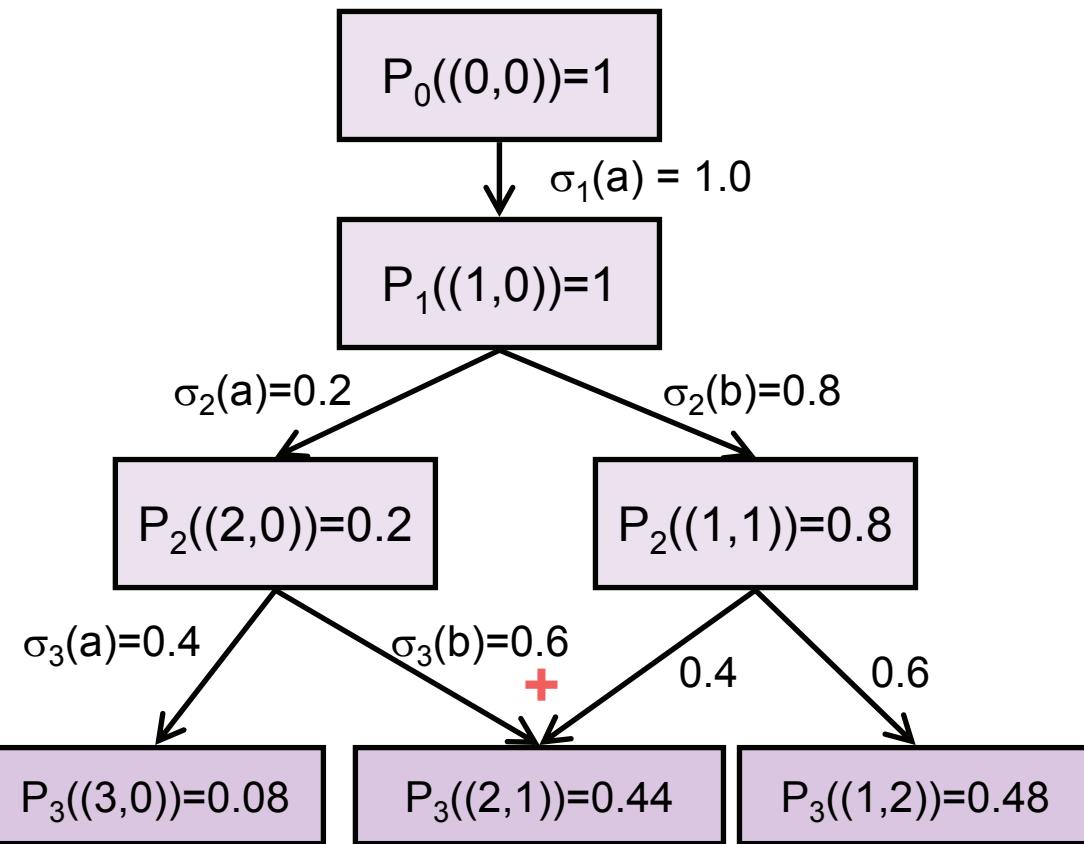
# AGG Example: 2 players

$$\begin{aligned}\sigma_1 &= (1, 0), \quad \sigma_2 = (0.2, 0.8), \\ \sigma_3 &= (0.4, 0.6), \quad \sigma_4 = (0.5, 0.5)\end{aligned}$$

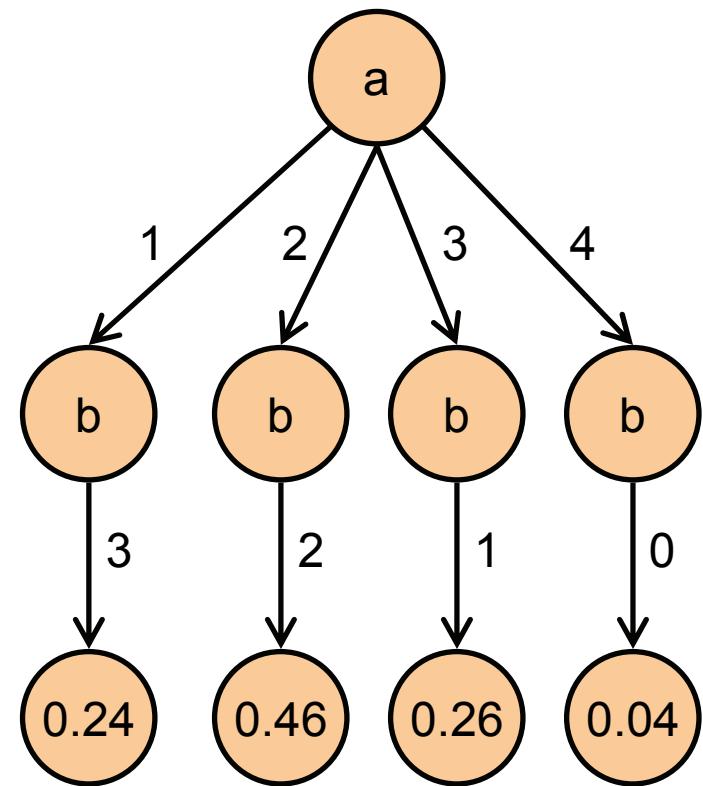
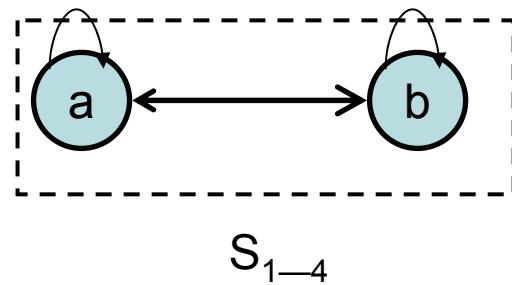
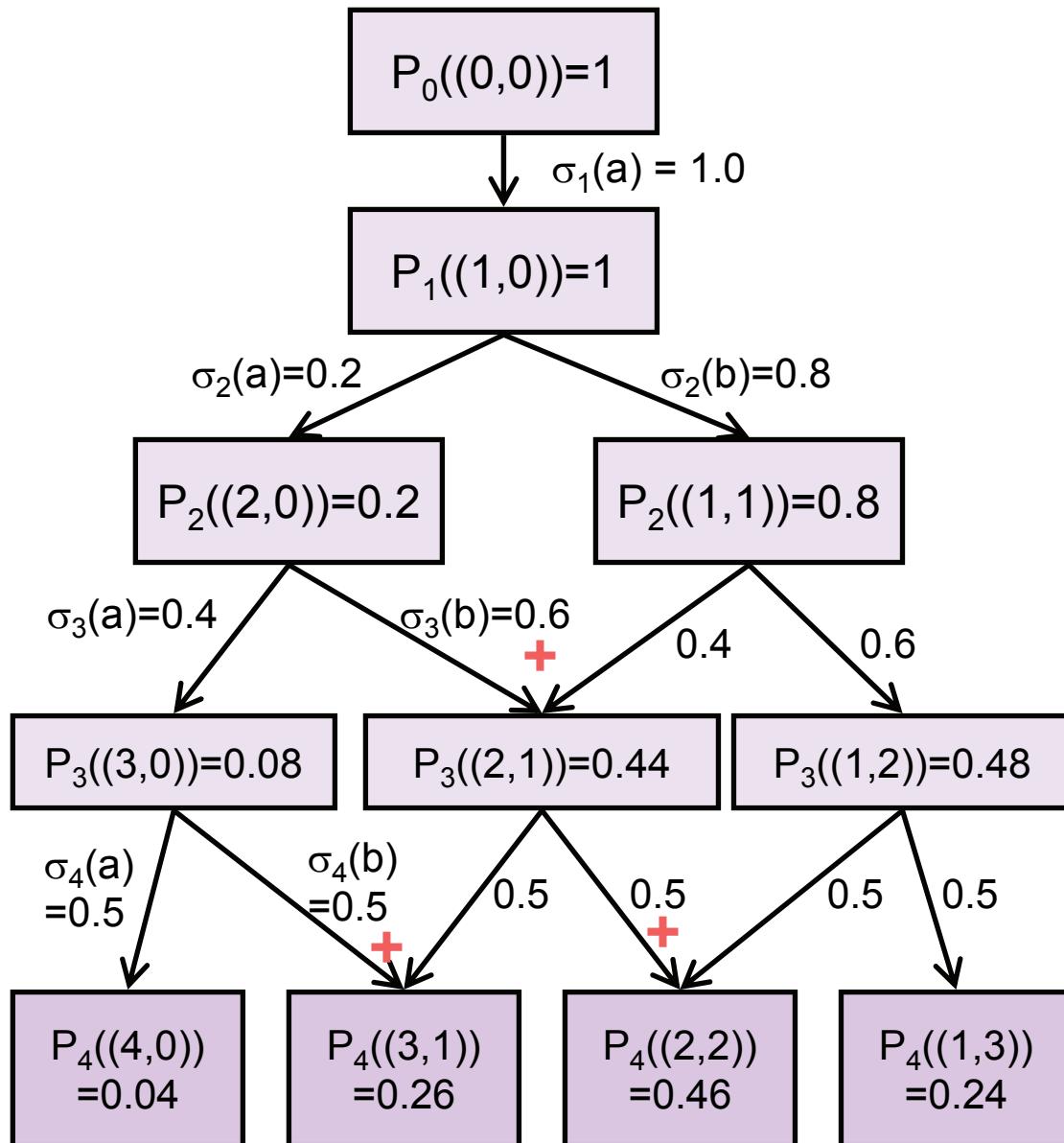


# AGG Example: 3 players

$$\sigma_1 = (1, 0), \sigma_2 = (0.2, 0.8), \\ \sigma_3 = (0.4, 0.6), \sigma_4 = (0.5, 0.5)$$



# AGG Example: 4 players



# Computing with AGGs: Complexity

**Theorem 1** *Given an AGG representation of a game, i's expected payoff  $V_{s_i}^i(\sigma_{-i})$  can be computed in time polynomial in the size of the representation. If  $\mathcal{I}$ , the in-degree of the action graph, is bounded by a constant,  $V_{s_i}^i(\sigma_{-i})$  can be computed in time polynomial in  $n$ .*

- Complexity of our approach:  
 $O(n^{\mathcal{I}} \text{poly}(n) \text{poly}(|S|))$
- Exponential speedup vs. standard approach:  
 $O(|S|^{n-1} \text{poly}(n) \text{poly}(|S|))$
- For graphical games encoded as AGGs, same exponential speedup as the special-purpose technique of [Blum, Shelton & Koller, 2002]

# AGGs with Function Nodes (AGGFNs)

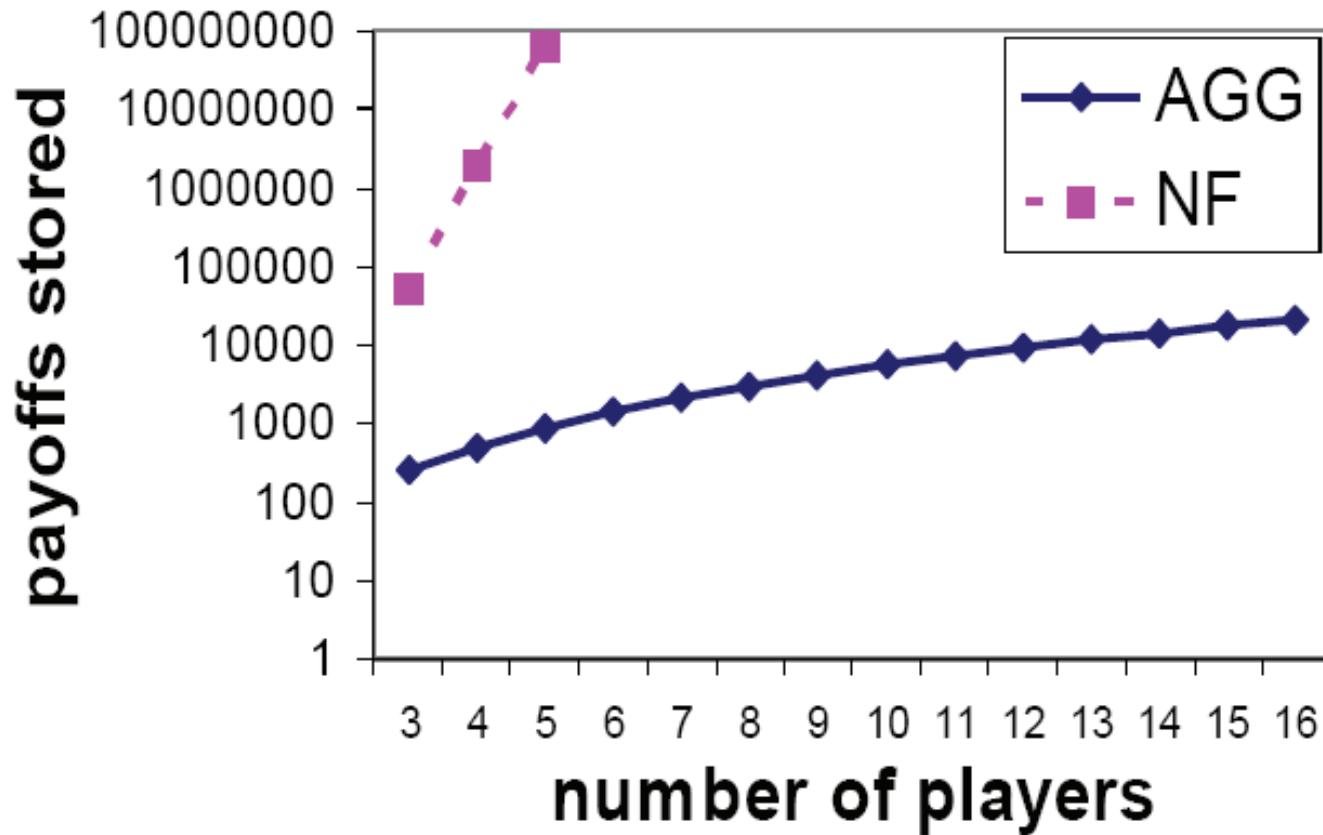
- Our dynamic programming algorithm does not work for **arbitrary** AGGFNs
  - players are no longer guaranteed to affect  $\mathbb{P}$  independently
- **Definition:** An AGGFN is **contribution-independent** (CI) if
  - all function nodes have only **action nodes** as their neighbors
  - there exists a **commutative and associative** operator  $*$ , and for each action node  $\bullet \in \blacklozenge$  an integer  $\bullet_*$ , such that given an action profile  $\bullet$ ,  
for all function nodes  $\square \in \blacklozenge$   $D(p) = \underset{i \in N : s_i \in \nu(p)}{*} w_{s_i}$
  - e.g., the coffee-shop game is CI, where  $*$  is sum and  $\forall \bullet \bullet_* = 1$
- **Theorem:** Our **dynamic programming algorithm works** with AGGFNs that are contribution-independent

# Overview on Action-Graph Games

1. Definition of AGGs and Examples
2. Analyzing and Extending the Representation
3. Computing with Games
4. Computing with AGGs
5. Experimental Results

# Experimental Results: Representation Size

*varying number of players*

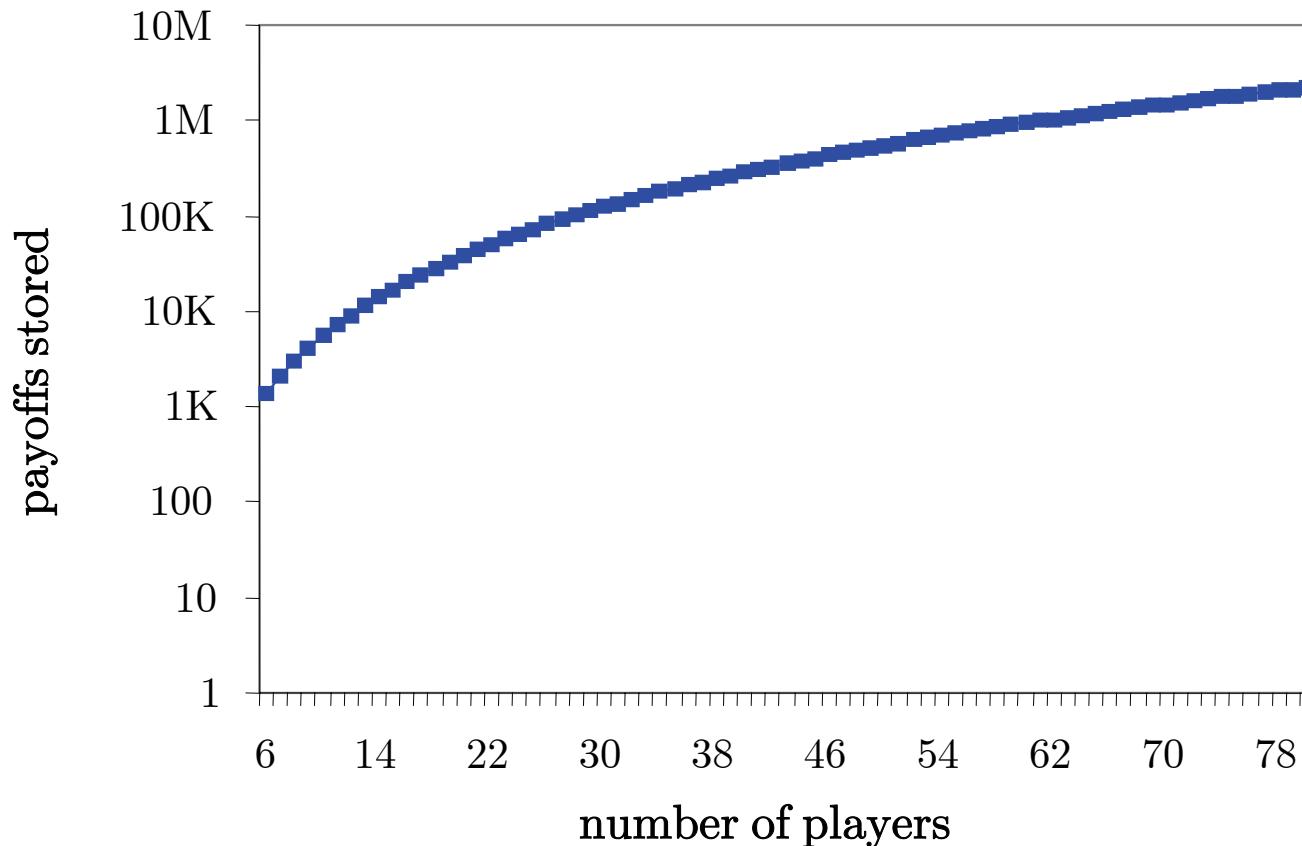


Coffee shop game,  $5 \times 5$  grid

*NF grows exponentially; AGG grows polynomially*

# Experimental Results: Representation Size

*varying number of players*

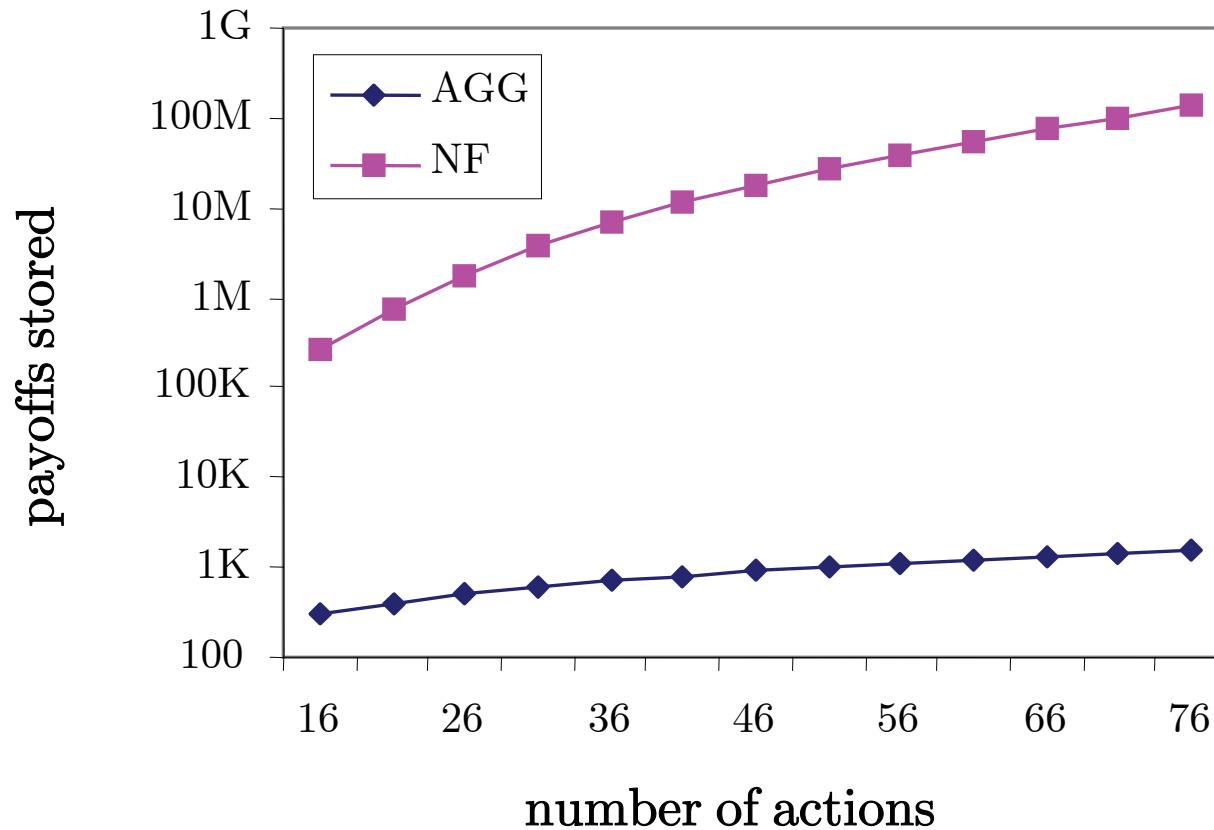


Coffee shop game,  $5 \times 5$  grid

*AGG grows polynomially*

# Experimental Results: Representation Size

*varying number of actions*

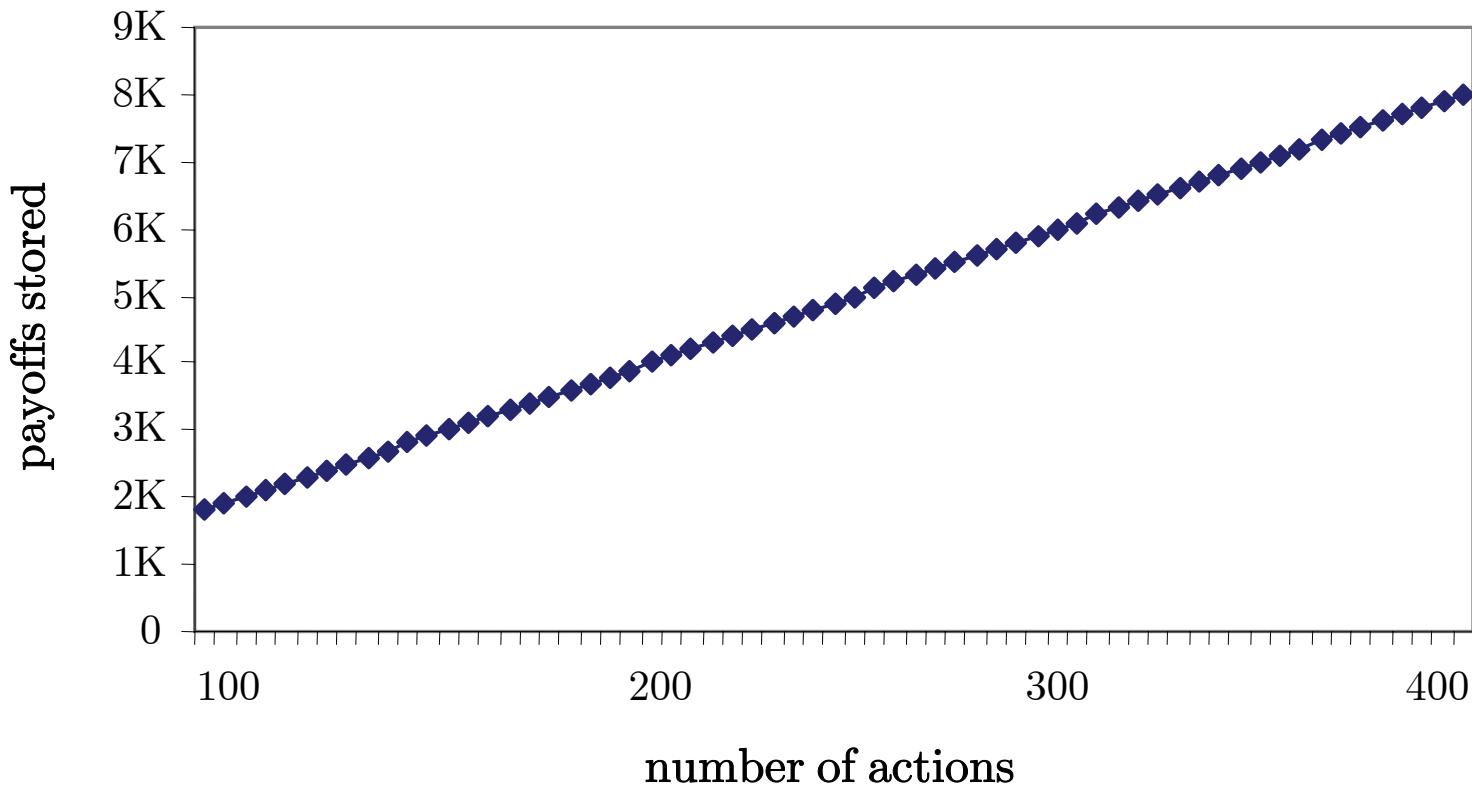


Coffee shop game, 4 players,  $\boxtimes \times 5$  grid

*AGG grows linearly, NF grows as a higher-order polynomial*

# Experimental Results: Representation Size

*varying number of actions*

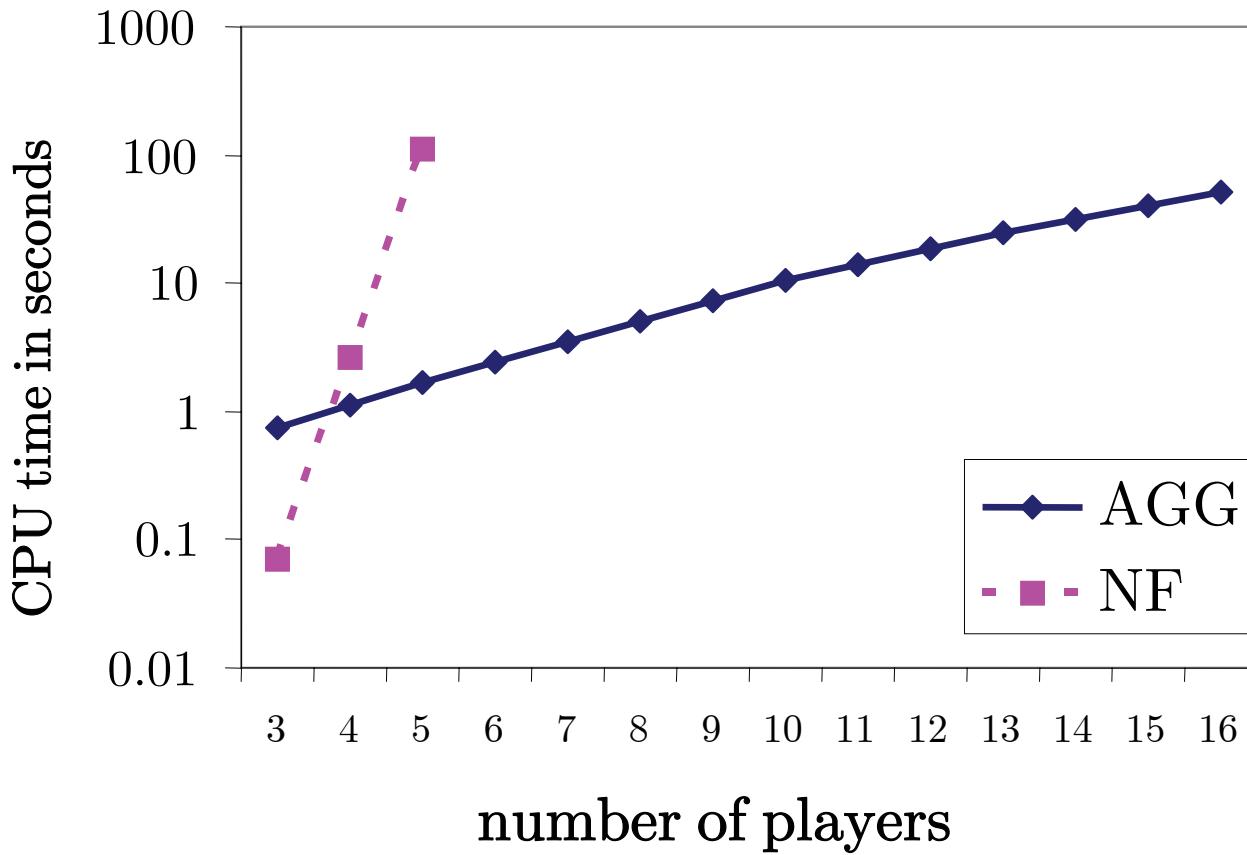


Coffee shop game, 4 players,  $\boxtimes \times 5$  grid

*AGG grows linearly*

# Experimental Results: Expected Payoff

*varying number of players*



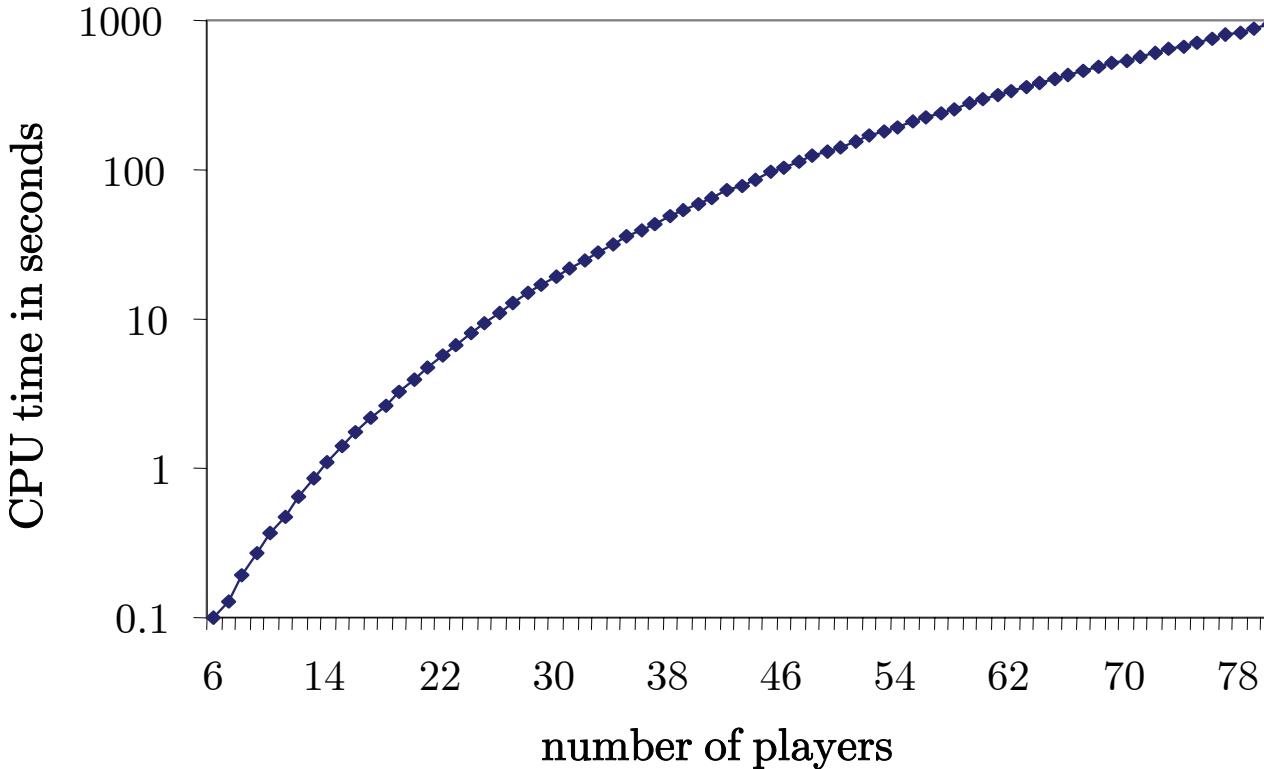
Coffee Shop Game,  $5 \times 5$  grid, AGG vs. GameTracer using NF

1000 random strategy profiles with full support

*AGG grows polynomially, NF grows exponentially*

# Experimental Results: Expected Payoff

*varying number of players*



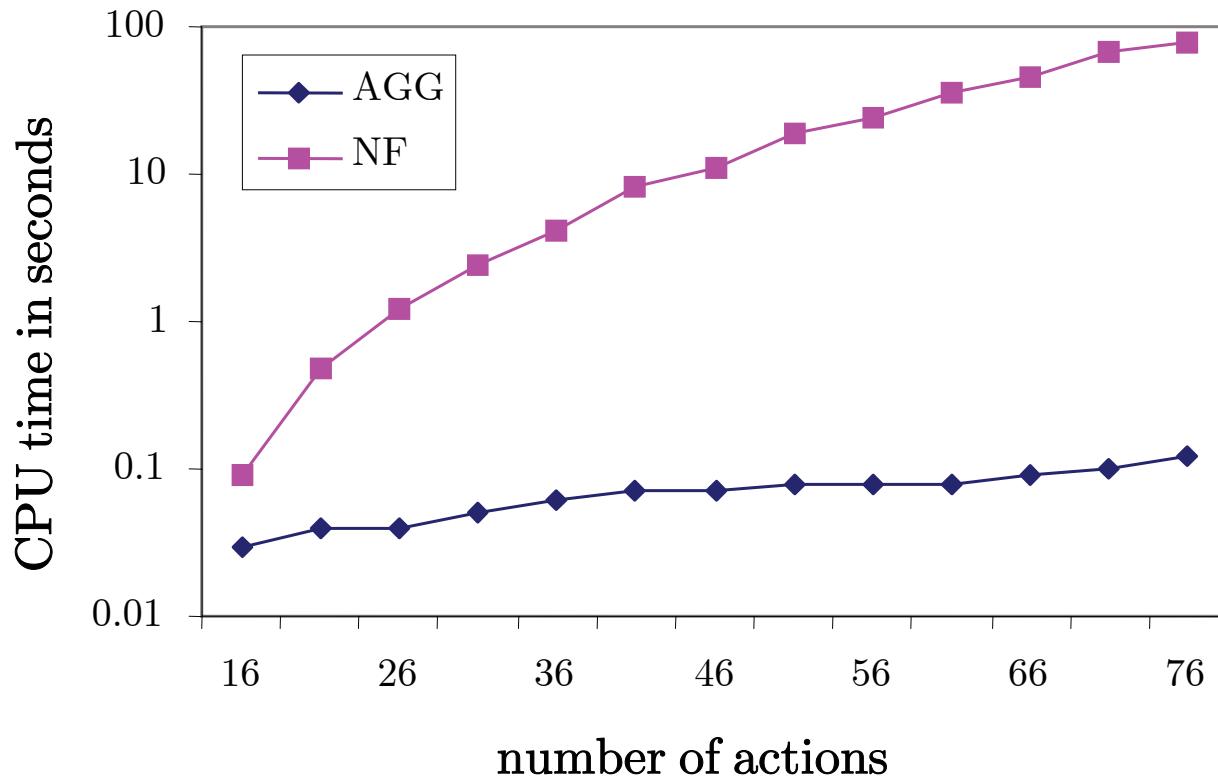
Coffee Shop Game,  $5 \times 5$  grid, AGG

1000 random strategy profiles with full support

*AGG grows polynomially*

# Experimental Results: Expected Payoff

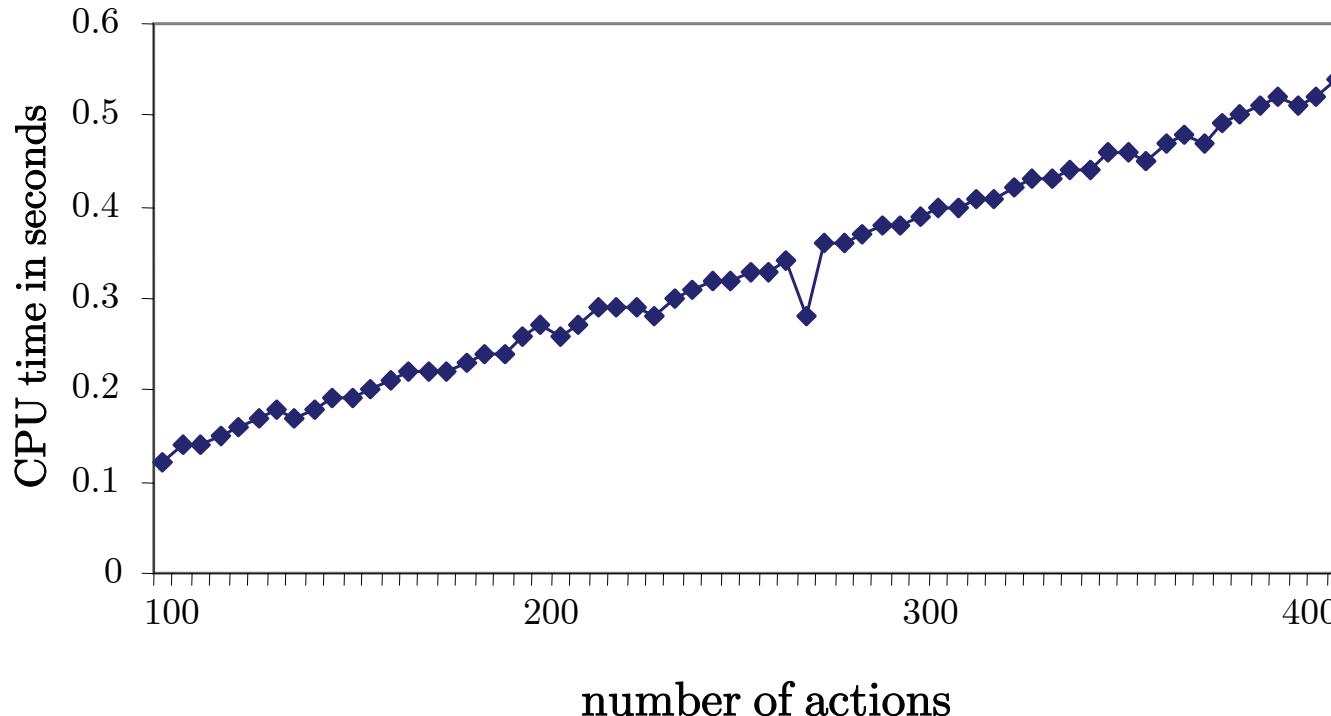
*varying number of actions*



Coffee Shop Game, 4 players,  $\boxtimes \times 5$  grid, AGG vs. GameTracer using  
1000 random strategy profiles with full support  
*AGG grows linearly, NF grows as higher-order polynomial*

# Experimental Results: Expected Payoff

*varying number of actions*

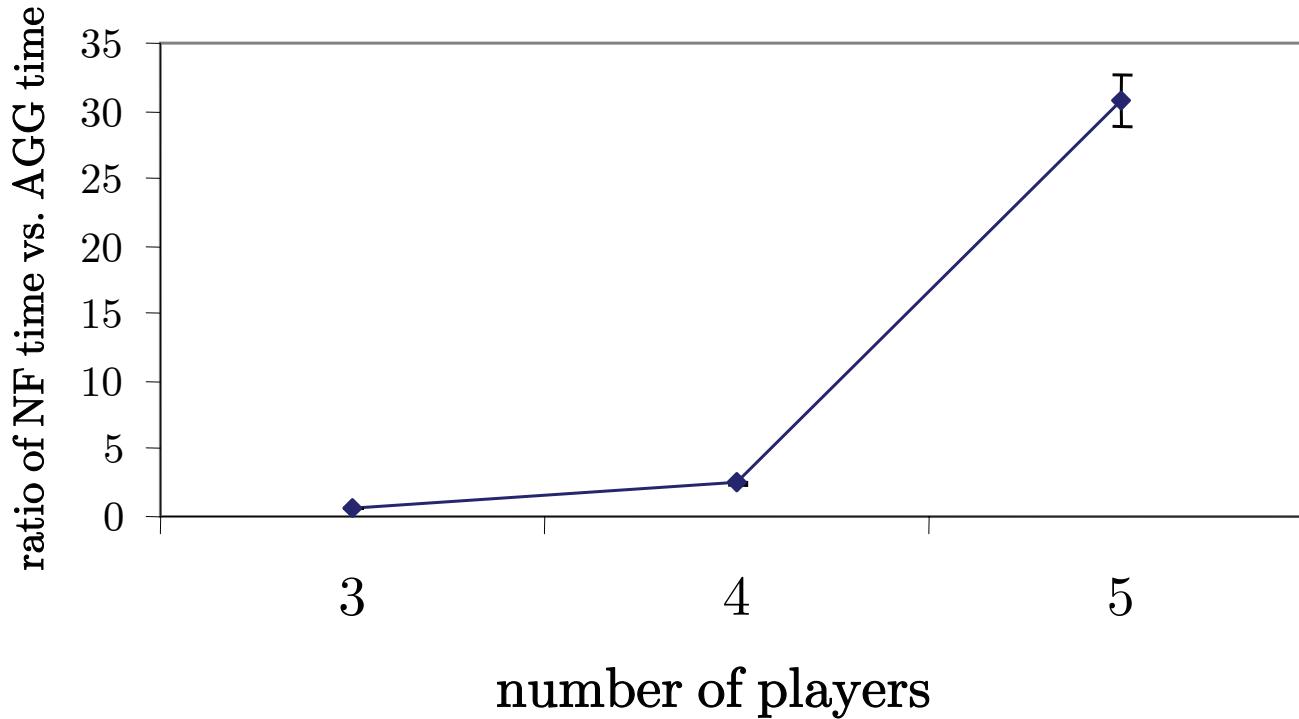


Coffee Shop Game, 4 players,  $\boxtimes \times 5$  grid, AGG vs. GameTracer using  
1000 random strategy profiles with full support

*AGG grows linearly*

# Experimental Results: Nash Equilibrium

*varying number of players*



Coffee Shop Game,  $4 \times 4$  grid, Govindan-Wilson Algorithm

Jacobians computed using AGGs vs. GameTracer using NF

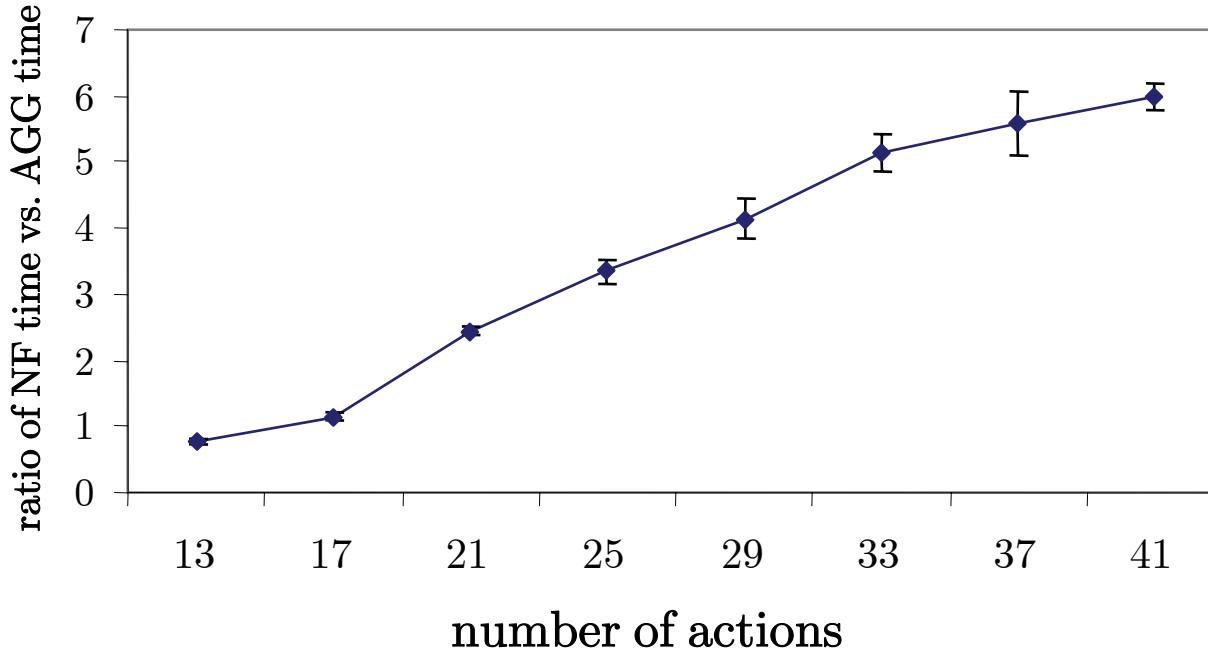
Exactly the same equilibria were found using both representations

Average across 10 initial perturbations; error bars indicate stdev

*As number of rows grows, AGG speedup increases roughly linearly*

# Experimental Results: Nash Equilibrium

*varying number of actions*



Coffee Shop Game,  $\boxtimes \times 4$  grid, Govindan-Wilson Algorithm

Jacobians computed using AGGs vs. GameTracer using NF

Exactly the same equilibria were found using both representations

Average across 10 initial perturbations; error bars indicate stdev

*As number of rows grows, AGG speedup increases roughly linearly*

# Coffee Shop Game: Example Equilibrium

1	1	-1.5	1
1	2	-1.5	2
-1.5	-6	-12.5	-6
2	-1.5	1.5	1.5

- Utility Function:  $5 - \boxed{x}^3 - \boxdot^2 - 0.5\boxtimes$   
 $\bowtie, \boxdot, \boxtimes$  are # of shops in same location, one block away, further away
- 5 players

# Coffee Shop Game: Example Equilibrium

0.5	-2	-2	-2
1.5	-2	1.5	1.5
-2	-13	-13	-13
1.5	-2	1.5	1.5

- Utility Function:  $5 - \boxed{x}^3 - \boxed{\square}^2 - 0.5 \ddot{x}$   
 $\ddot{x}, \boxed{x}, \boxed{\square}$  are # of shops in same location, one block away, further away
- 6 players

# Coffee Shop Game: Example Equilibrium

0.5	-2.5	-2.5	-2.5
0.5	-7	0.5	0.5
-2.5	-13.5	-13.5	-13.5
1	-7	0.5	0.5

- Utility Function:  $5 - \square^3 - \square^2 - 0.5 \#$   
where  $\square$ ,  $\square$ ,  $\#$  are # of shops in same location, one block away, further away
- 7 players

# Coffee Shop Game: Example Equilibrium

0.5	-2.5	-2.5	-2.5
0.5	-7	0.5	0.5
-2.5	-13.5	-13.5	-13.5
1	-7	0.5	0.5

- Utility Function:  $5 - \square^3 - \square^2 - 0.5 \#$   
as  $\square$ ,  $\square$ ,  $\#$  are # of shops in same location, one block away, further away
- 8 players; one chooses not to participate

# Conclusions

## Action-Graph Games

- **Fully-expressive** compact representation of games exhibiting context-specific independence and/or strict independence
- Permit **efficient computation** of expected utility under a mixed strategy, which allows efficient computation of e.g., best response, Nash equilibrium, etc.
- **Generalizes** graphical games
- Experimentally: much **faster** than the normal form



# Job Market Game

