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 Watson strolls past the site of a soon-to-open Starbucks coffee shop in Toronto's Leslieville area.


VANCOUVER

FRIDAY MAY 5 2006

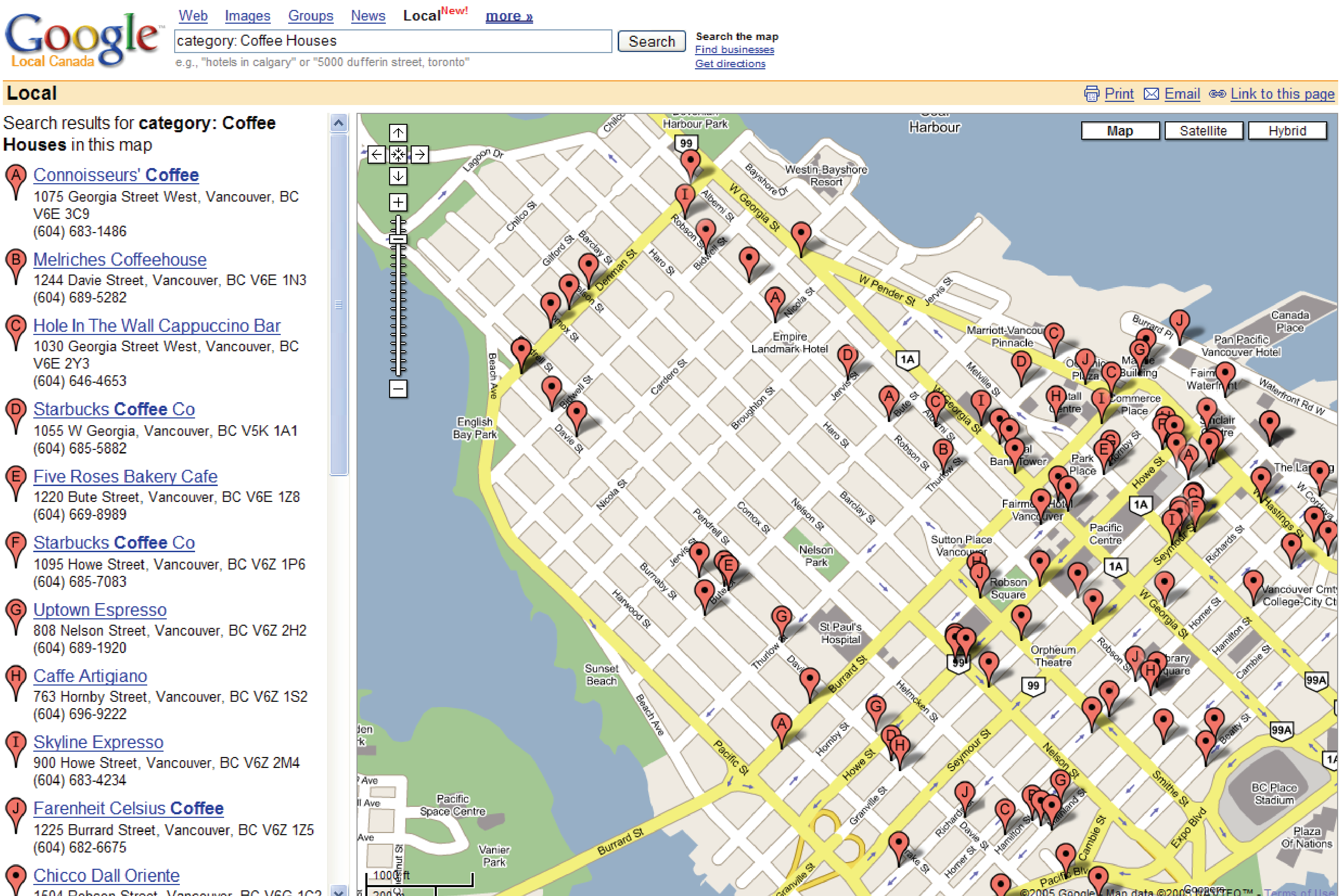
VOL. 2 N°26

24 hours

24hrs.ca

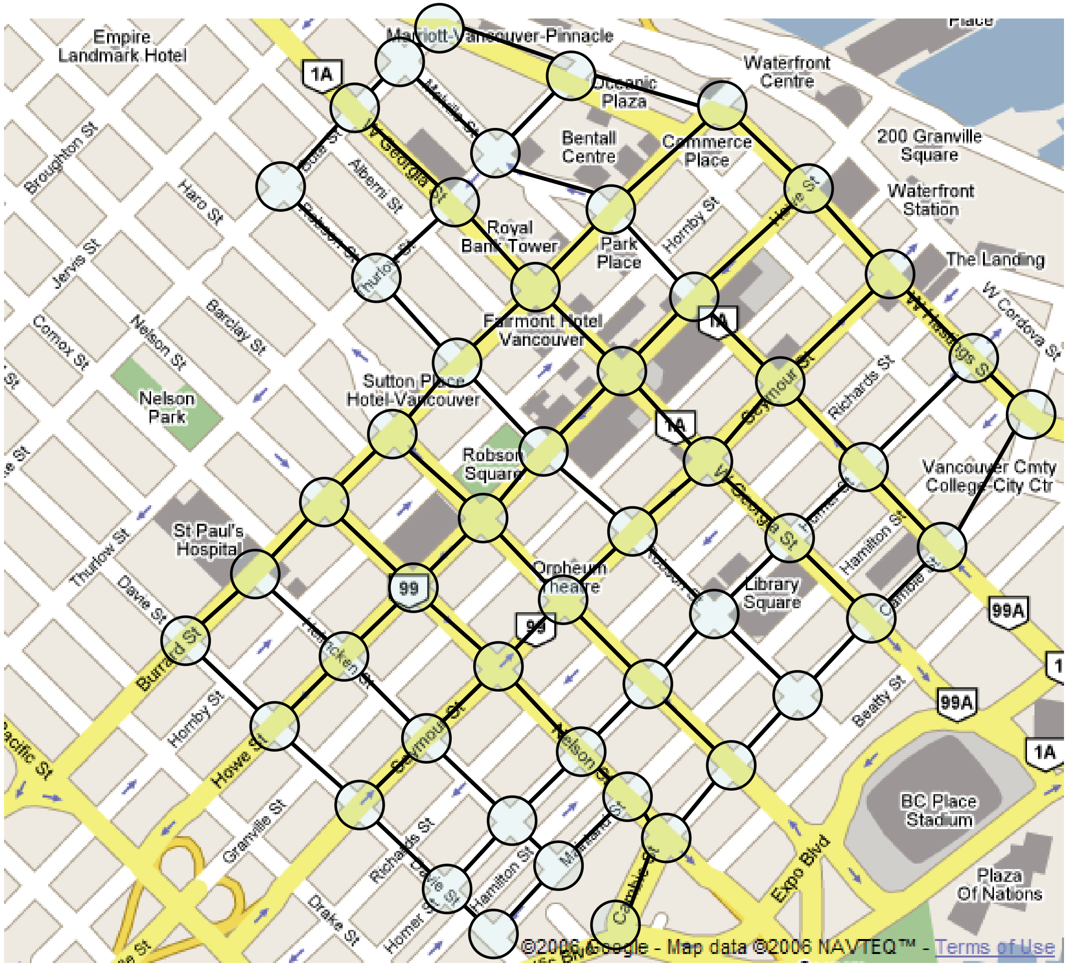
COFFEE WARS

The Coffee Shop Problem



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 $A = (A_1, \ldots, A_n)$ 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^{th} *element* $D(j)$ *is interpreted as the number of agents who chose the* j^{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* Δ*.*

Formal Definitions

Definition 3 (neighborhood relation) *Given a graph having a set of nodes* $\mathcal{A}$ *and edges* E*, define the* **neighborhood relation** *as* $\nu : \mathcal{A} \to 2^{\mathcal{A}}$*, with* $\nu(i) = \{j | (j, i) \in E\}$.

Define a ***configuration over a node's neighborhood***, written as $D^{(\nu(j))} \in \Delta^{(\nu(j))}$, as the elements of D that correspond to the actions $\nu(j)$.

Definition 4 *An* **action-graph game (AGG)** *is a tuple* (N, A, G, u)*, where:*

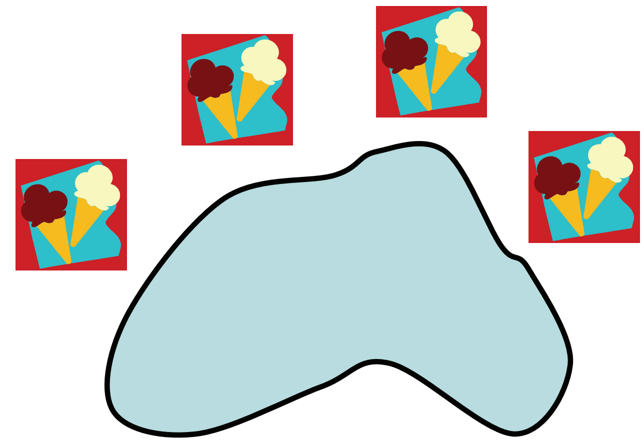
- N *is the set of agents;*
- $A = (A_1, \ldots, A_n)$*, where* A_i *is the set of actions available to agent* i*;*
- $G = (\mathcal{A}, E)$ *is an action graph, where* $\mathcal{A} = \bigcup_{i \in N} A_i$ *is the set of distinct actions;*
- $u = (u_1, \ldots, u_{|\mathcal{A}|})$, $u_j : \Delta^{(\nu(j))} \to \mathbb{R}$.

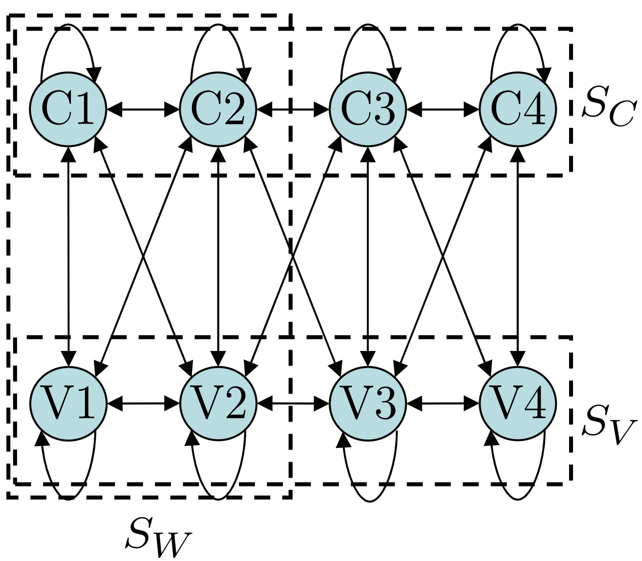
Elaborated Ice Cream Vendor Problem

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:

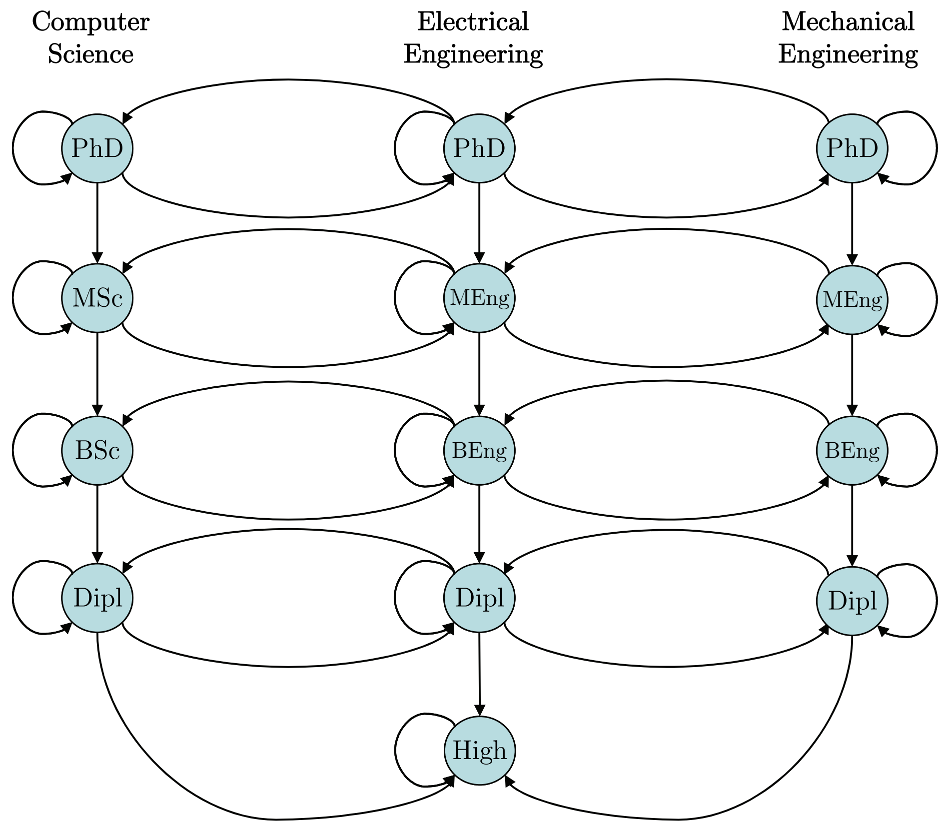
- graph structure independent of # agents
- overlapping action sets
- context-specific independence without strict independence

The Job Market Problem

Each player chooses a level of training

Players' utilities are the sum of:

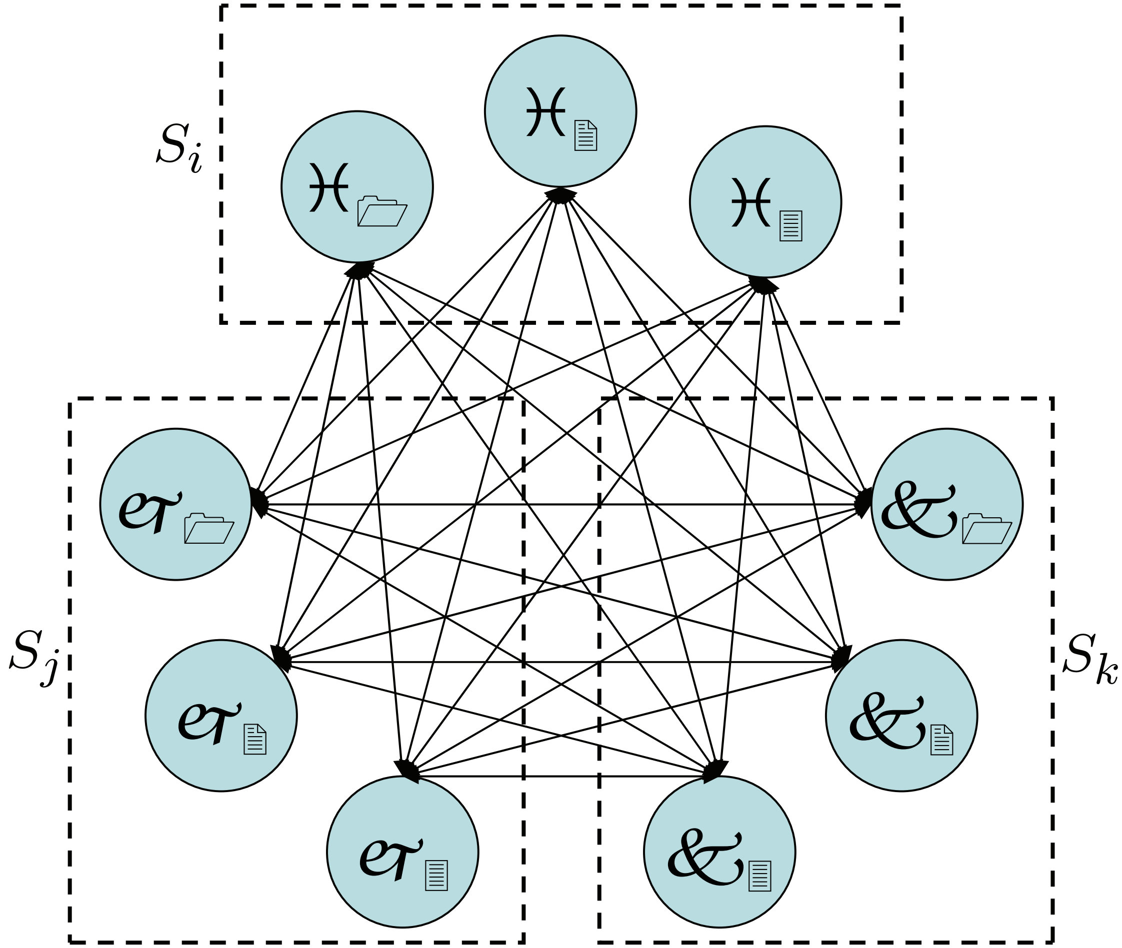
- a constant cost:
  - difficulty; tuition; foregone wages
- a variable reward, depending on:
  - How many jobs prefer workers with this training, and how desirable are the jobs?
  - How many other jobs are willing to take such workers as a second choice, and how good are these jobs?
    - Employers 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

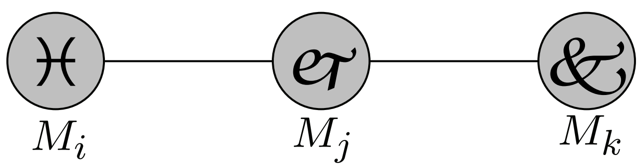
2. **Anonymity:**
   - multiple action profiles yield the same configuration

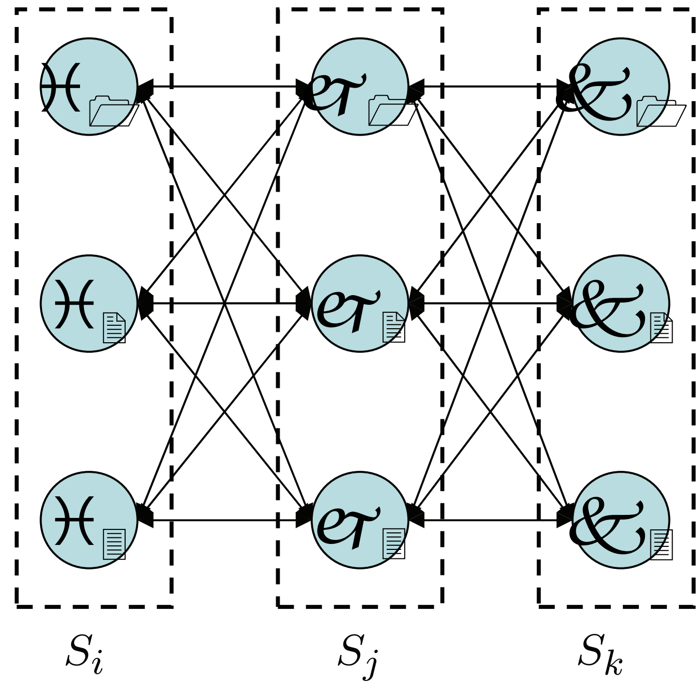
Size of the AGG representation

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

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

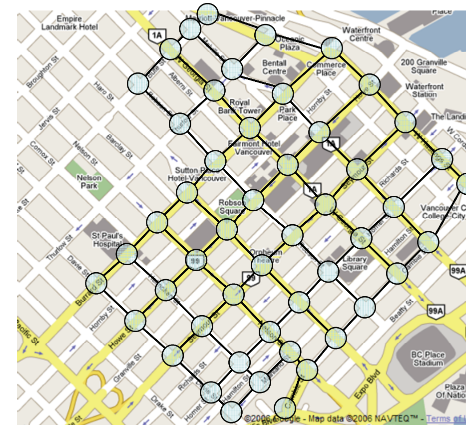
Graphical Games are Compact as AGGs



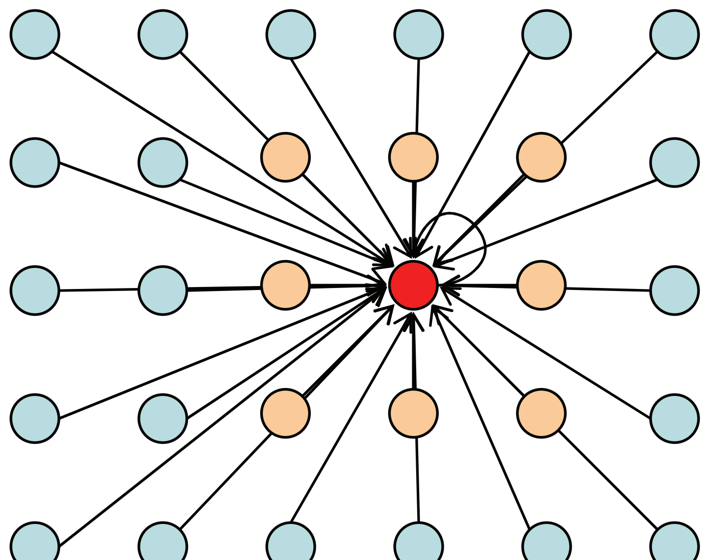


GG	AGG
Agent node	Action set box
Edge	Bipartite graphs between action sets
Local game matrix	Node utility function

The Coffee Shop Problem Revisited



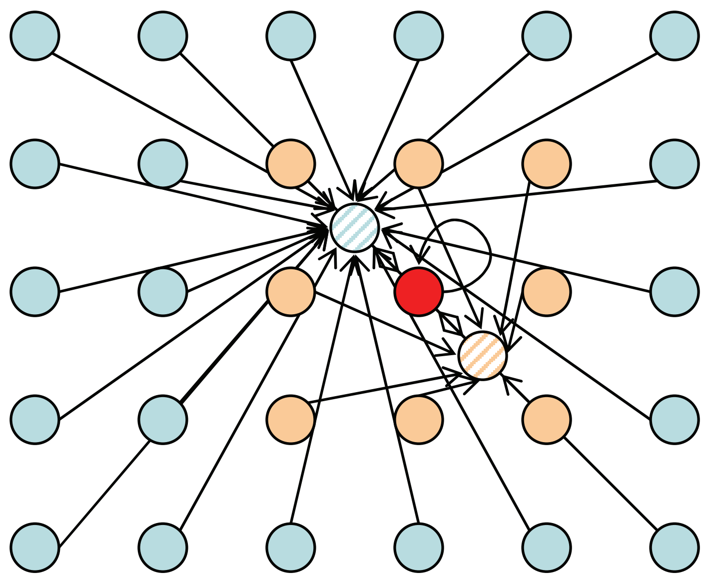
- What if utility also depends on total # shops?
- Now action graph has in-degree $|\mathcal{A}|$
  - NF & Graphical Game representations: ⚐($|\mathcal{A}|$^☠)
  - AGG representation: ⚐(☠ $^{|\mathcal{A}|}$)
  - when $|\mathcal{A}|$ is held constant, the AGG representation is polynomial in ☠
    - but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



6 × 5 Coffee Shop Problem: projected action graph at the red node

Function Nodes

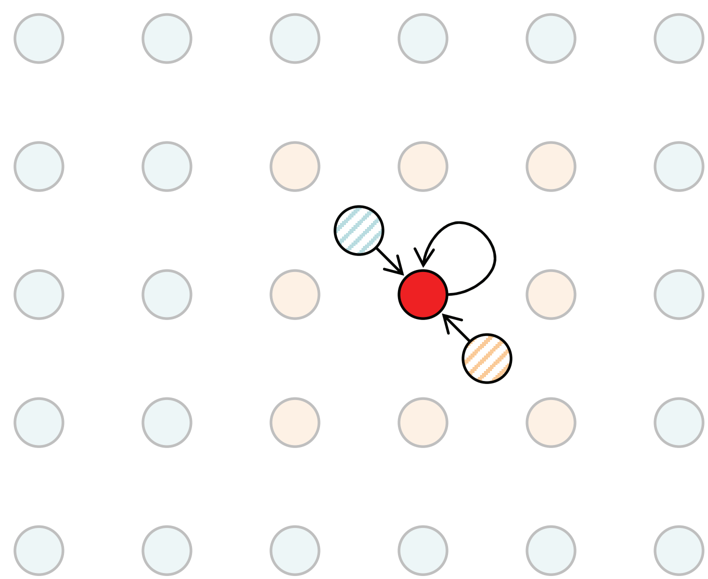
- To exploit this structure, introduce function nodes:
  - The "configuration" of a function node is a (given) function of the configuration of its neighbors: 👎(□) = ↗$_{□}$(👎(ν(□)))
- Coffee-shop example: for each action node ♦, introduce:
  - One function node with adjacent actions as neighbours
    - 👎(□'$_{♦}$) = total # of shops in surrounding nodes
  - Similarly, a function node with non-adjacent actions as neighbours



6 × 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 ⚐(☠^{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 ♓ for playing action $\blacklozenge_{\text{♓}}$,
if other agents play according to mixed-strategy profile $\boldsymbol{\sigma}_{-\text{♓}}$:

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

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

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

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

Computing with Games

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

- 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)

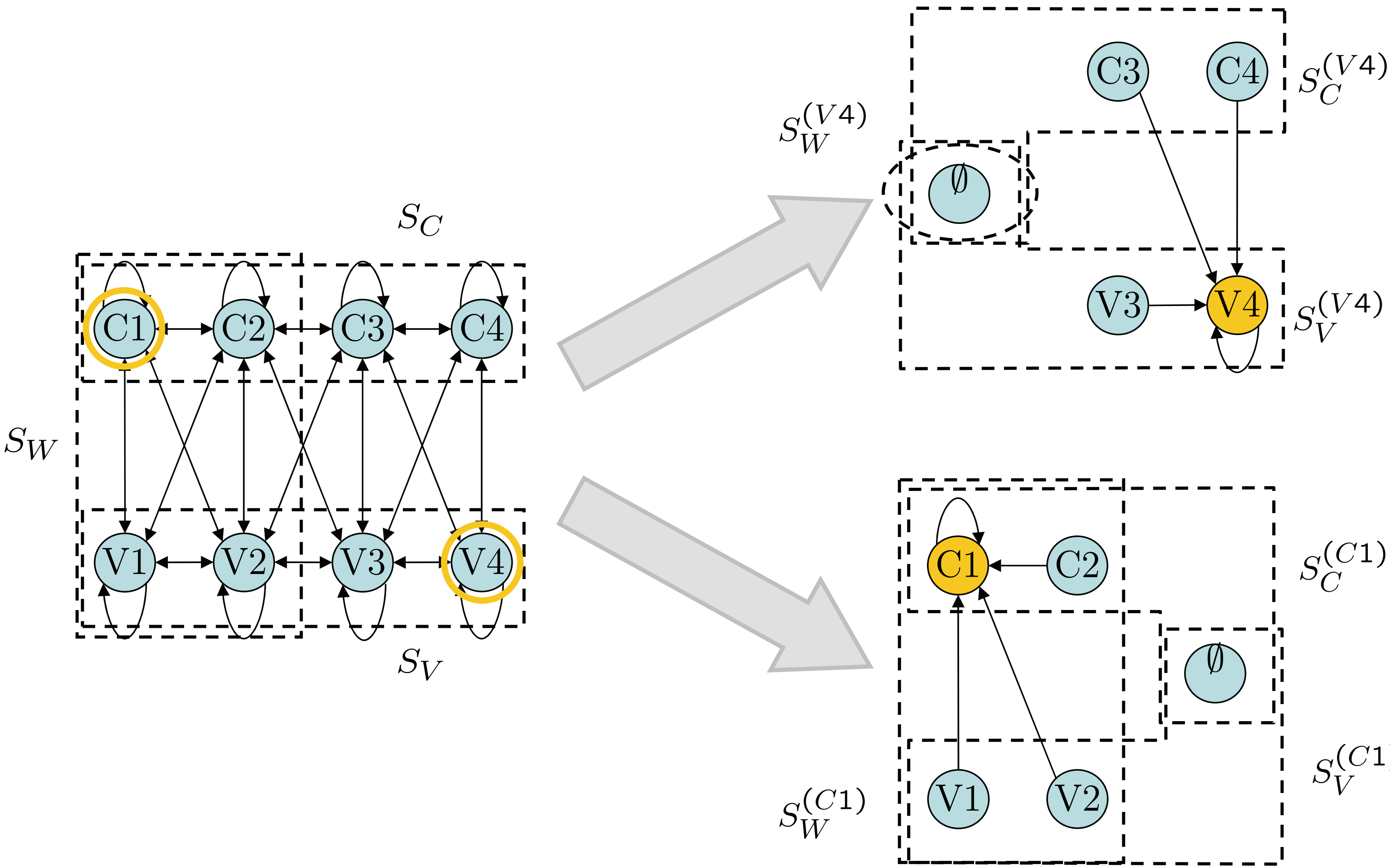
$$V^{i}_{s_i}(\sigma_{-i}) = \sum_{\mathbf{s}_{-i} \in \mathbf{S}_{-i}} u_i(s_i, \mathbf{s}_{-i})\, Pr(\mathbf{s}_{-i} | \sigma_{-i})$$

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

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

$$V^i_{s_i}(\overline{\sigma}) = \sum_{\overline{\mathbf{s}}^{(s_i)} \in \overline{\mathbf{S}}^{(s_i)}} u^{s_i}\left(\mathcal{D}(s_i, \overline{\mathbf{s}}^{(s_i)})\right) Pr\left(\overline{\mathbf{s}}^{(s_i)} | \overline{\sigma}^{(s_i)}\right)$$

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

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

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

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

Computing with AGGs: Anonymity

- Writing in terms of the configuration captures anonymity

$$V^i_{s_i}(\overline{\sigma}) = \sum_{\overline{D}^{(s_i)} \in \overline{\Delta}^{(s_i)}} u^{s_i}\left(\mathcal{D}\left(s_i, \overline{D}^{(s_i)}\right)\right) Pr\left(\overline{D}^{(s_i)} | \overline{\sigma}^{(s_i)}\right)$$

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

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

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

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

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

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

- 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

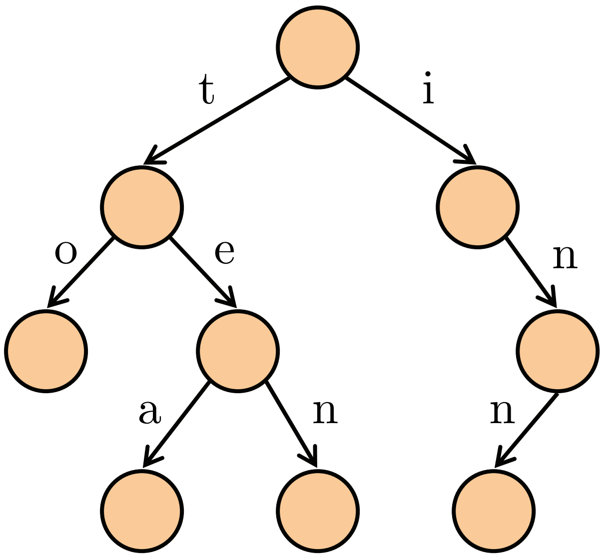
Dynamic Programming

- A ray of hope: note that
  - the players' mixed strategies are independent
    - i.e. σ is a product probability distribution
  - each player affects the configuration 👎 independently

- We can use dynamic programming to compute the probability of a configuration:
  - base case: zero agents and the mixed strategy σ_0:
    - $\Delta_0 = \{$👎$_0\}$
    - 👎$_0 = \{0, \ldots, 0\}$
    - 🏳$_0($👎$_0) = 1$
  - then add agents one by one:
    - $\Delta_{\&}$: the set of configurations that can be built by adding any action the support of player &'s mixed strategy to any configuration from $\Delta_{\&\text{-}1}$
    - $$P_k(D_k) = \sum_{\substack{(D_{k-1}, s_k), \\ \mathcal{D}(D_{k-1}, s_k) = D_k}} \sigma_k(s_k) \cdot P_{k-1}(D_{k-1})$$

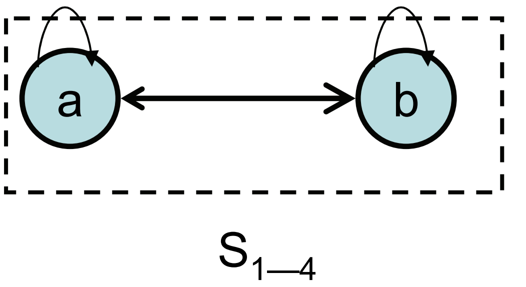
Dynamic Programming

- 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

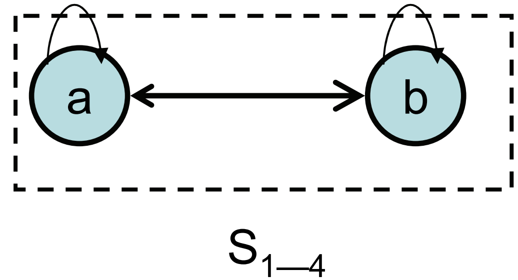


- Example game:
  - 4 players, 2 actions

- Compute joint probability distribution σ where $\sigma_1=(1, 0)$, $\sigma_2=(0.2, 0.8)$, $\sigma_3=(0.4, 0.6)$, $\sigma_4=(0.5, 0.5)$

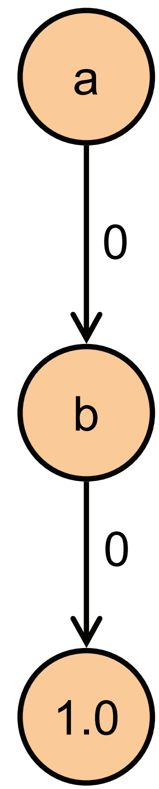
AGG Example: 0 players

- Example game:
  - 4 players, 2 actions



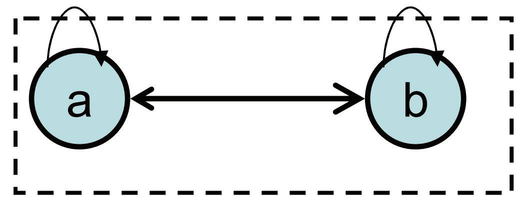
- Compute joint probability distribution σ where $\sigma_1=(1, 0)$, $\sigma_2=(0.2, 0.8)$, $\sigma_3=(0.4, 0.6)$, $\sigma_4=(0.5, 0.5)$

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

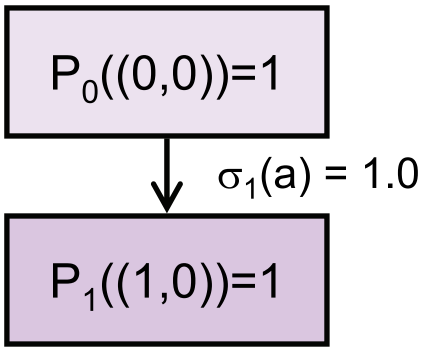


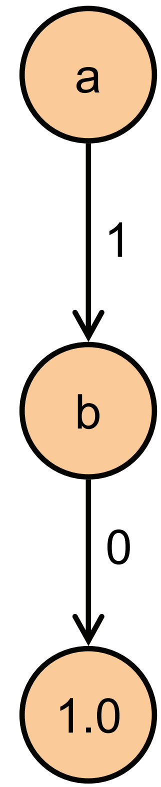
AGG Example: 1 player

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



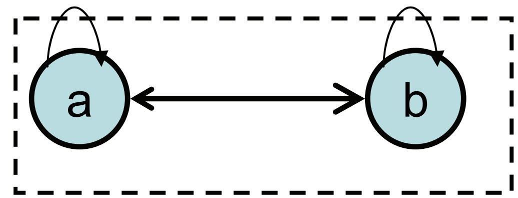
$S_{1—4}$



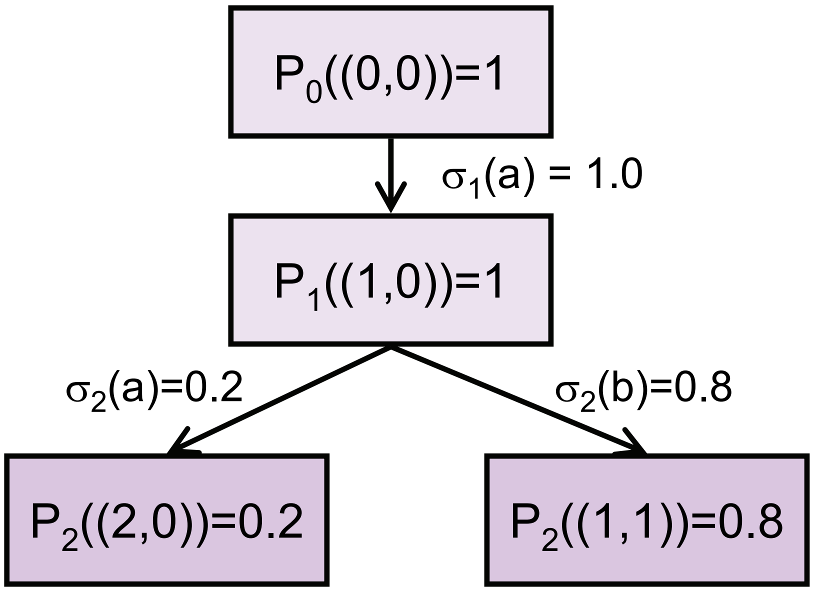


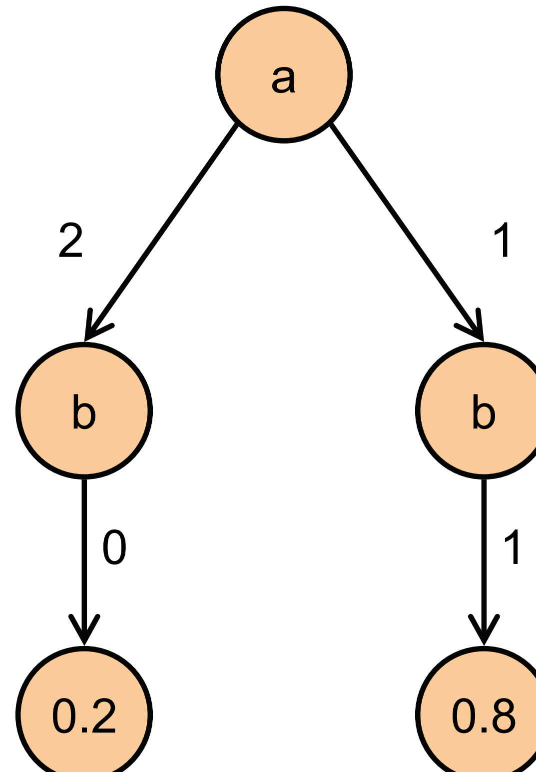
AGG Example: 2 players

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



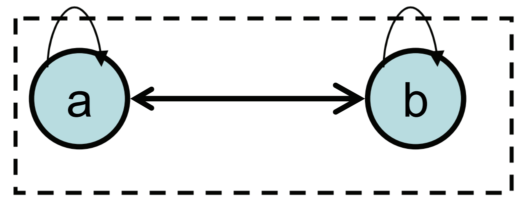
$S_{1—4}$



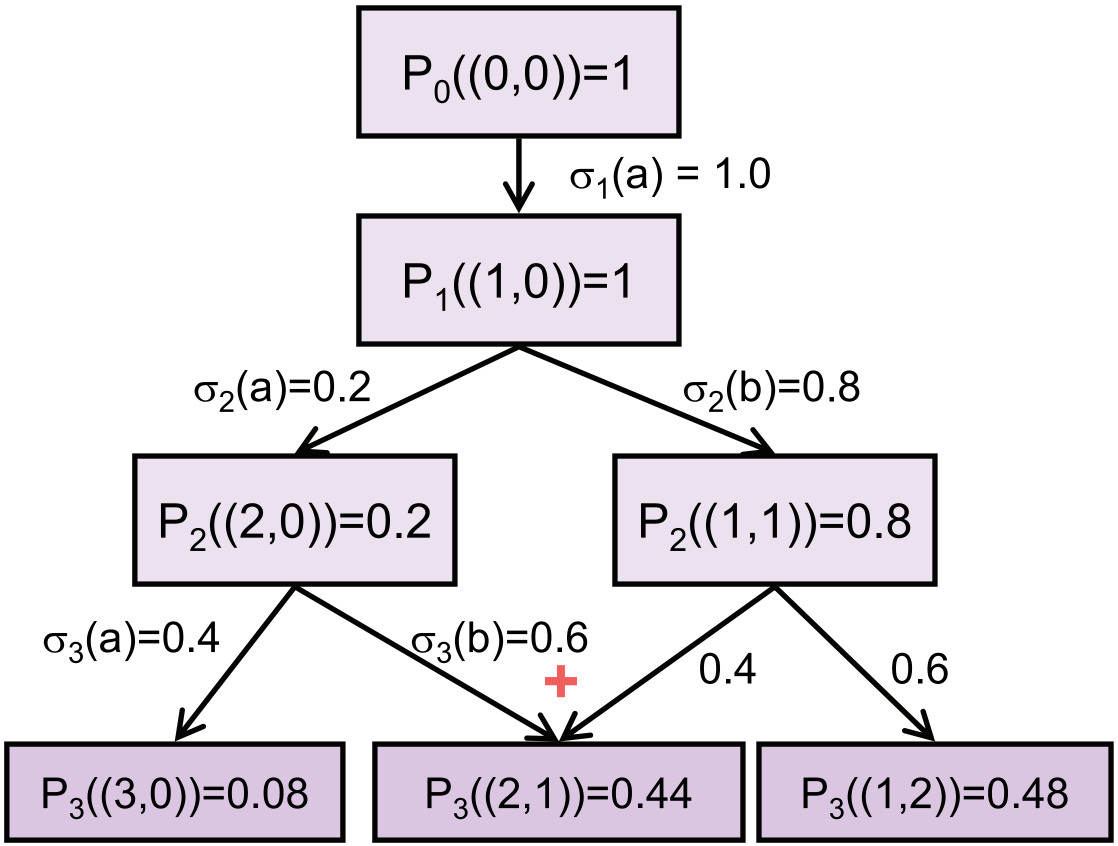


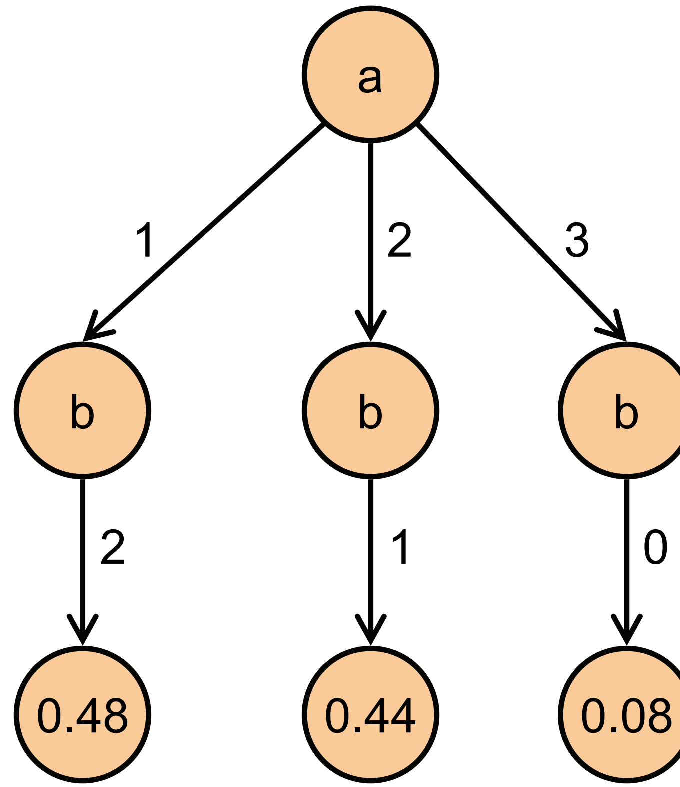
AGG Example: 3 players

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

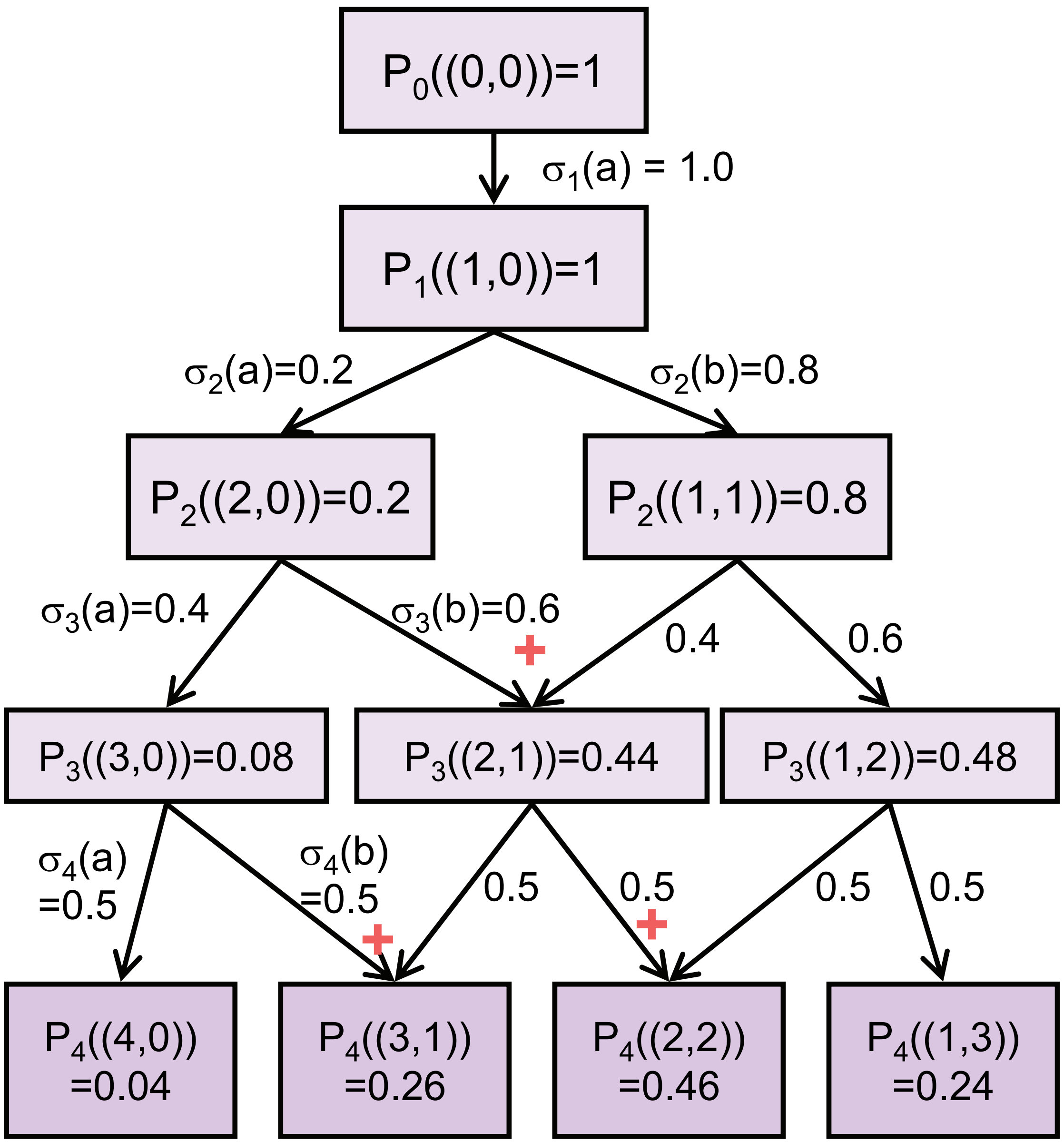


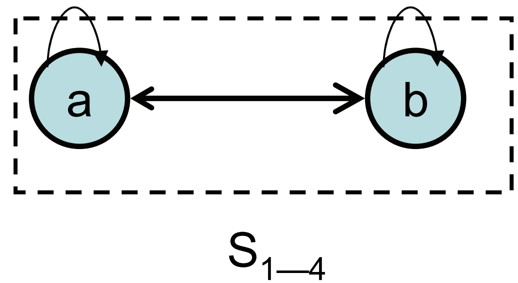
S_{1-4}

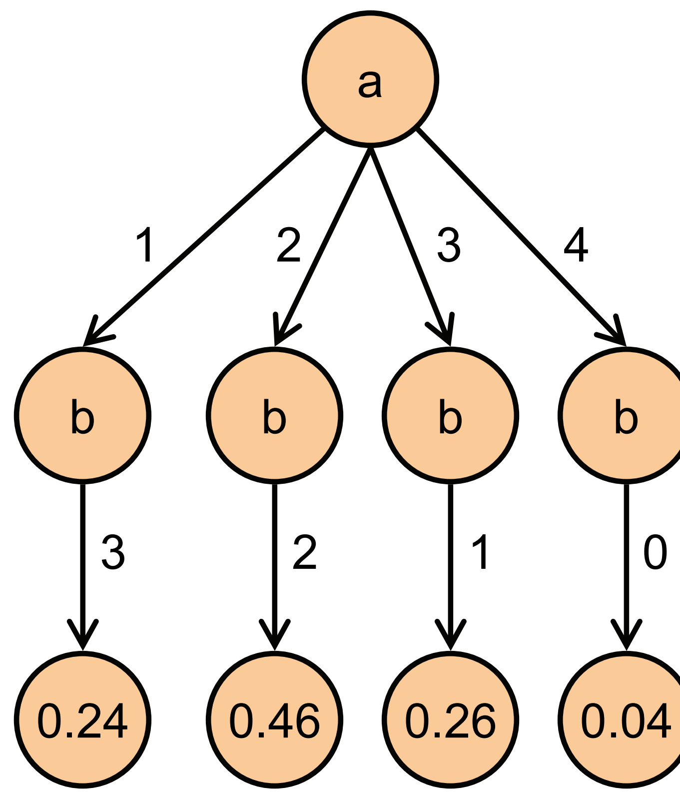




AGG Example: 4 players







Computing with AGGs: Complexity

Theorem 1 *Given an AGG representation of a game, i's expected payoff* $V^i_{s_i}(\sigma_{-i})$ *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,* $V^i_{s_i}(\sigma_{-i})$ *can be computed in time polynomial in* n*.*

- Complexity of our approach:

  $$O\left(n^{\mathcal{I}} poly(n) poly(|S|)\right)$$

- Exponential speedup vs. standard approach:

  $$O\left(|S|^{n-1} poly(n) poly(|S|)\right)$$

- 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 👎 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 ♦ ∈ ♠ an integer ◆♦, such that given an action profile ♦, for all function nodes ◘ ∈ ⚐ $D(p) = \mathop{\ast}\limits_{i \in N \,:\, s_i \in \nu(p)} w_{s_i}$
  - e.g., the coffee-shop game is CI, where * is sum and ∀♦ ◆♦ = 1

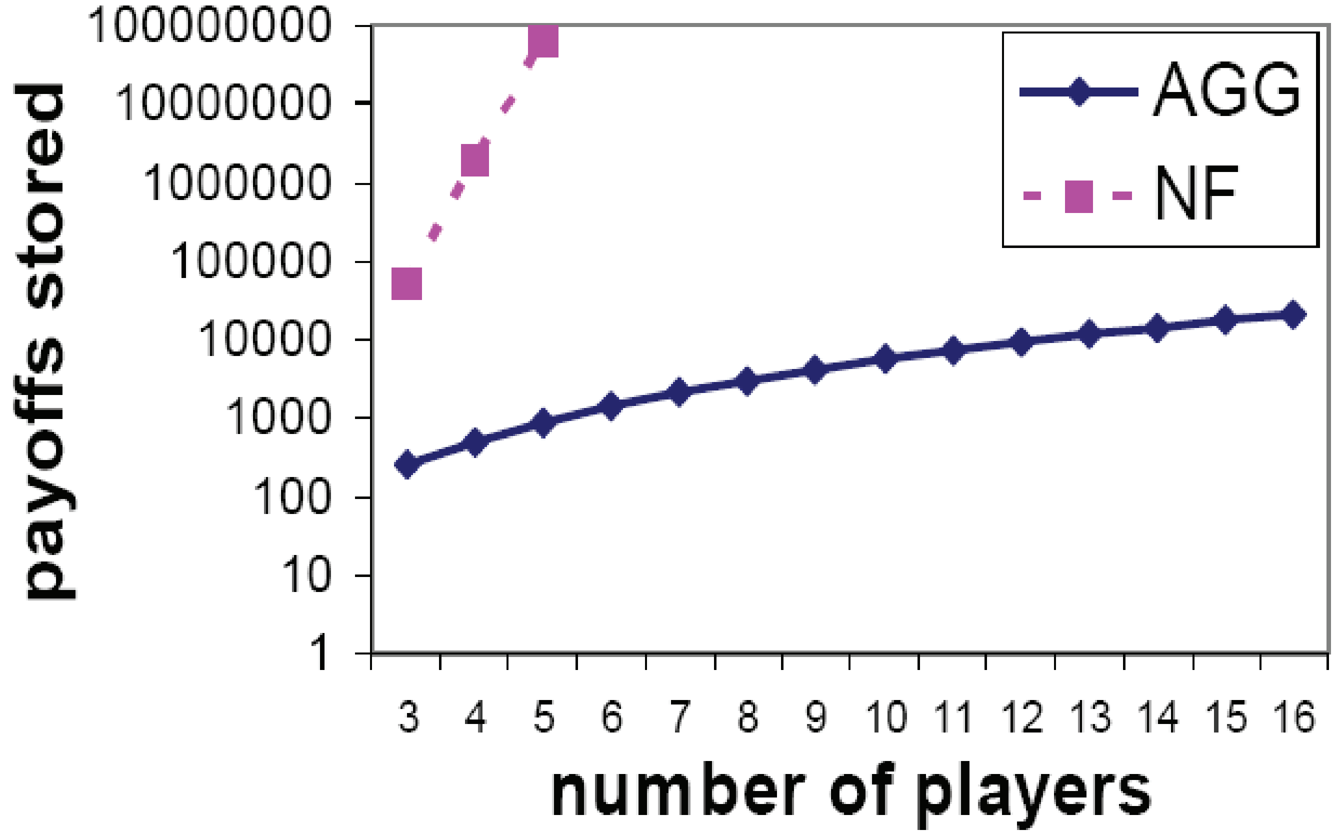
- **Theorem:** Our dynamic programming algorithm works with AGGFNs that are contribution-independent

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***

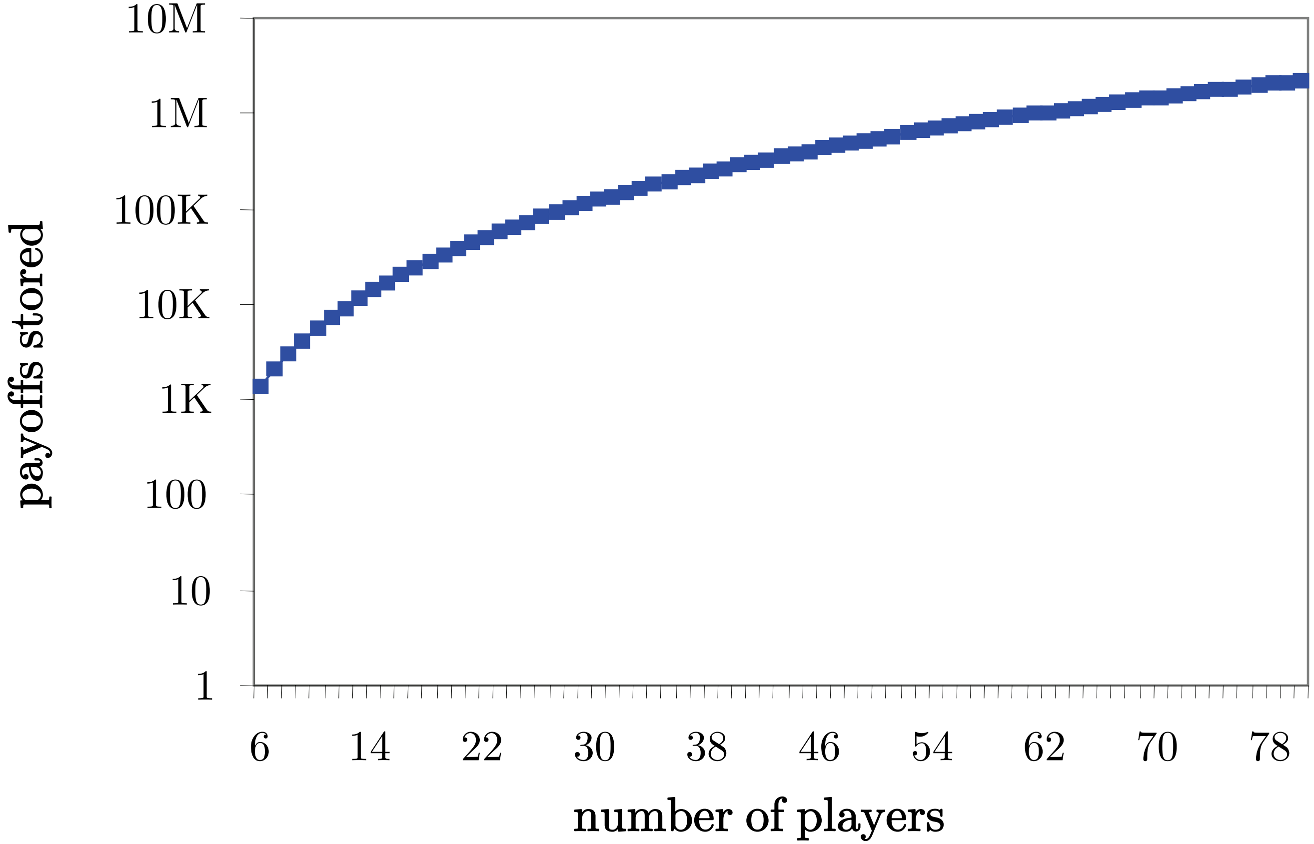


Coffee shop game, 5×5 grid

NF grows exponentially; AGG grows polynomially

Experimental Results: Representation Size

varying number of players

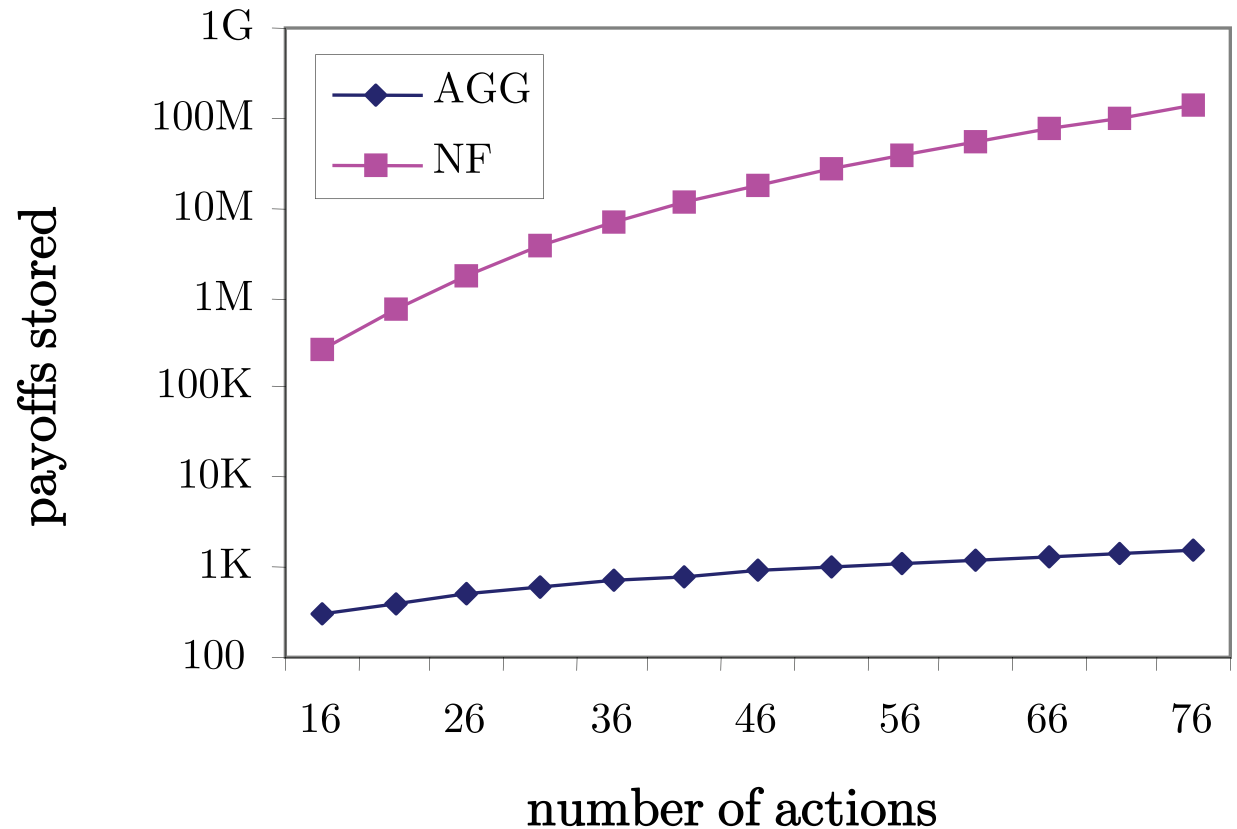


Coffee shop game, 5 × 5 grid

AGG grows polynomially

Experimental Results: Representation Size

varying ***number of actions***

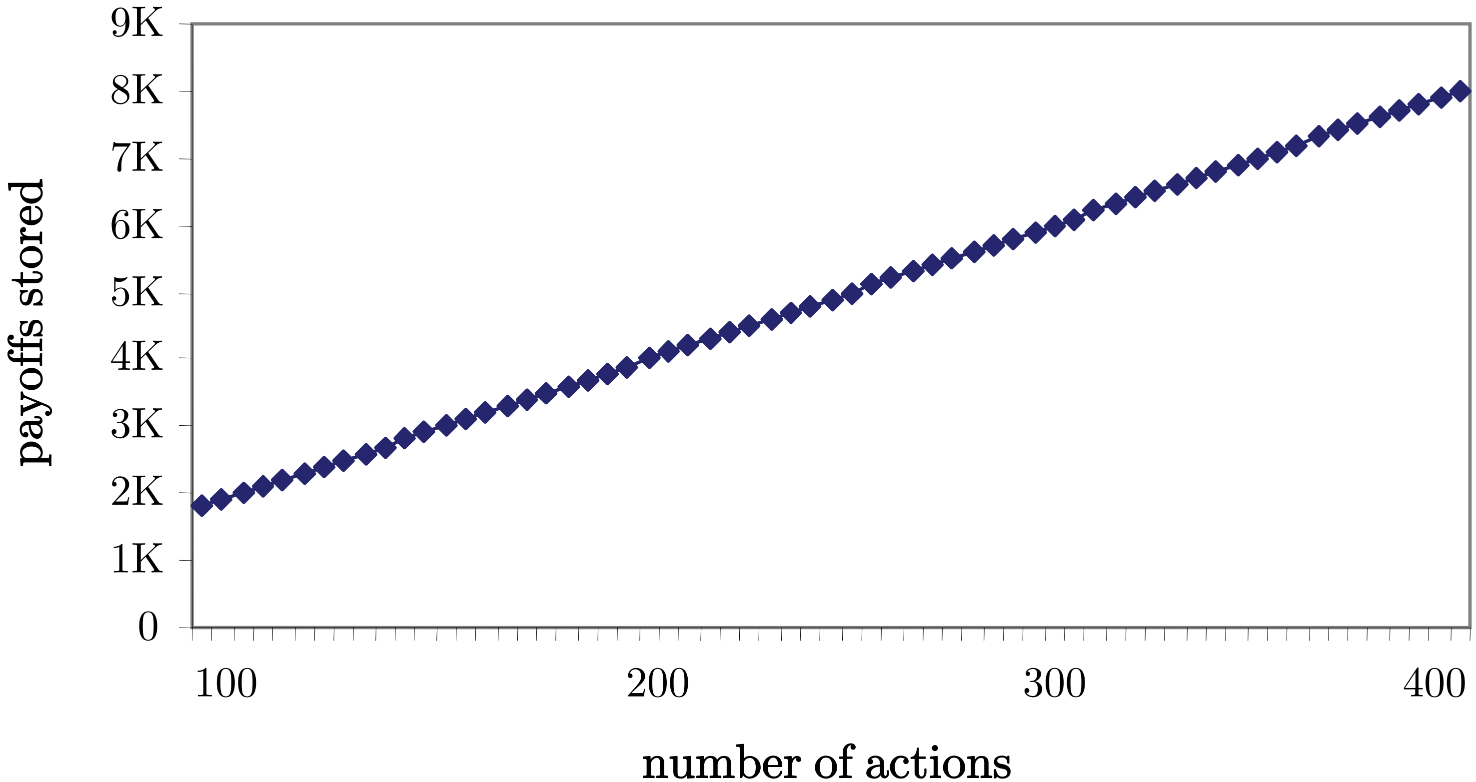


Coffee shop game, 4 players, ☒ × 5 grid

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

Experimental Results: Representation Size

*varying **number of actions***

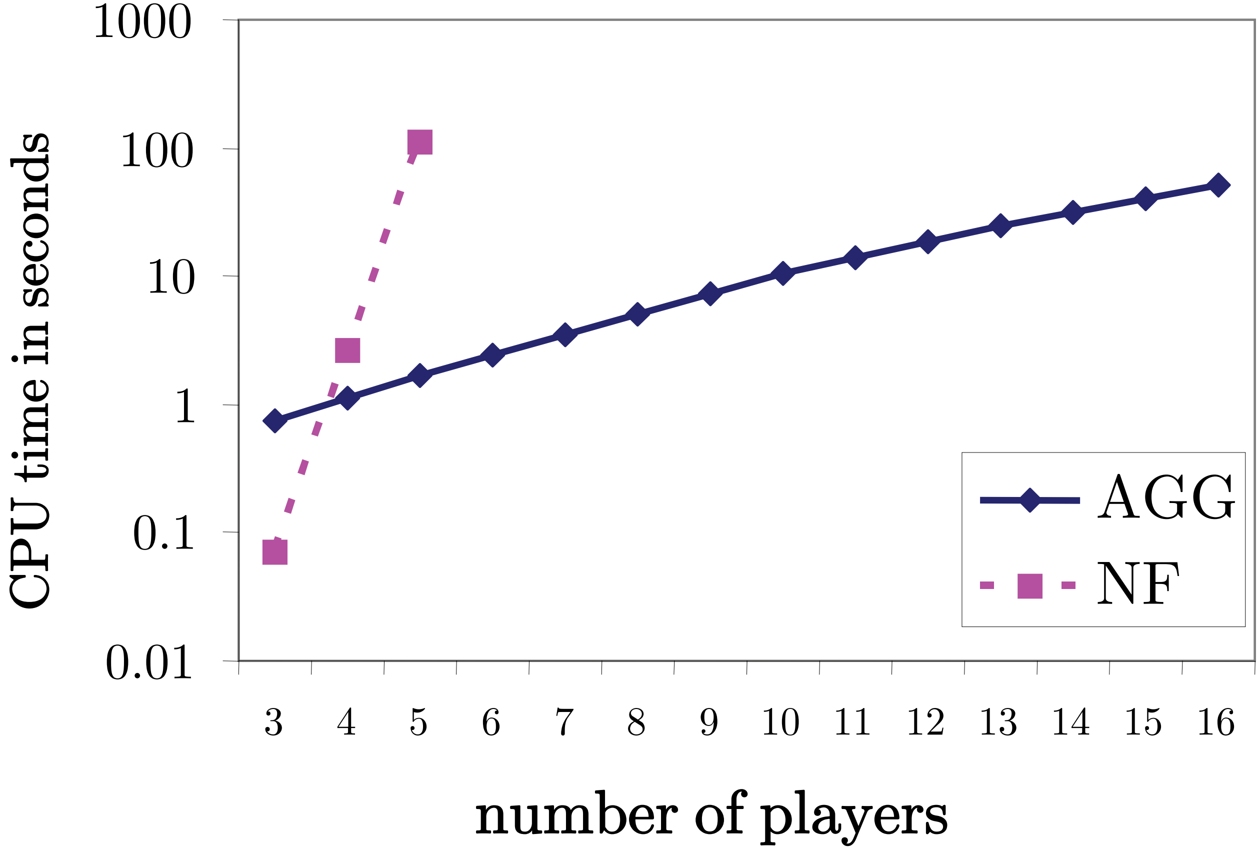


Coffee shop game, 4 players, ☒ × 5 grid

AGG grows linearly

Experimental Results: Expected Payoff

varying number of players



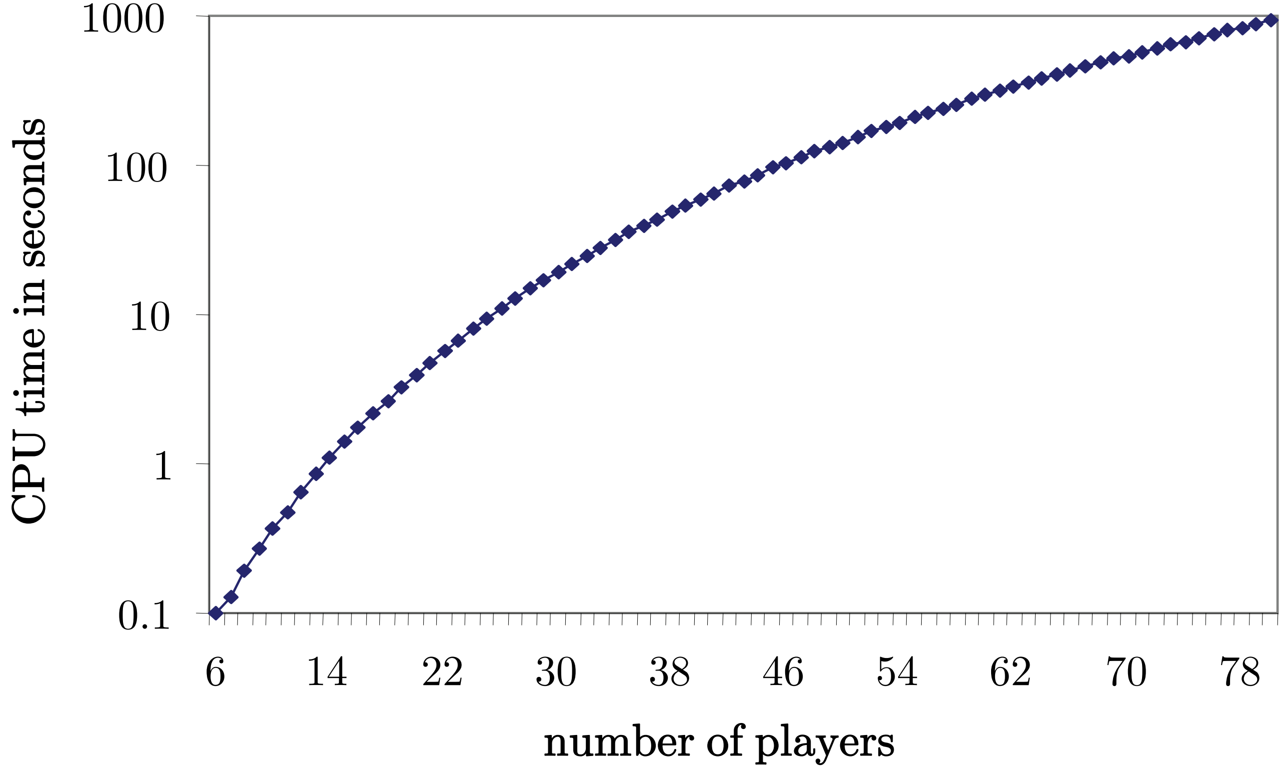
Coffee Shop Game, 5 × 5 grid, AGG vs. GameTracer using NF

1000 random strategy profiles with full support

AGG grows polynomially, NF grows exponentially

Experimental Results: Expected Payoff

varying ***number of players***



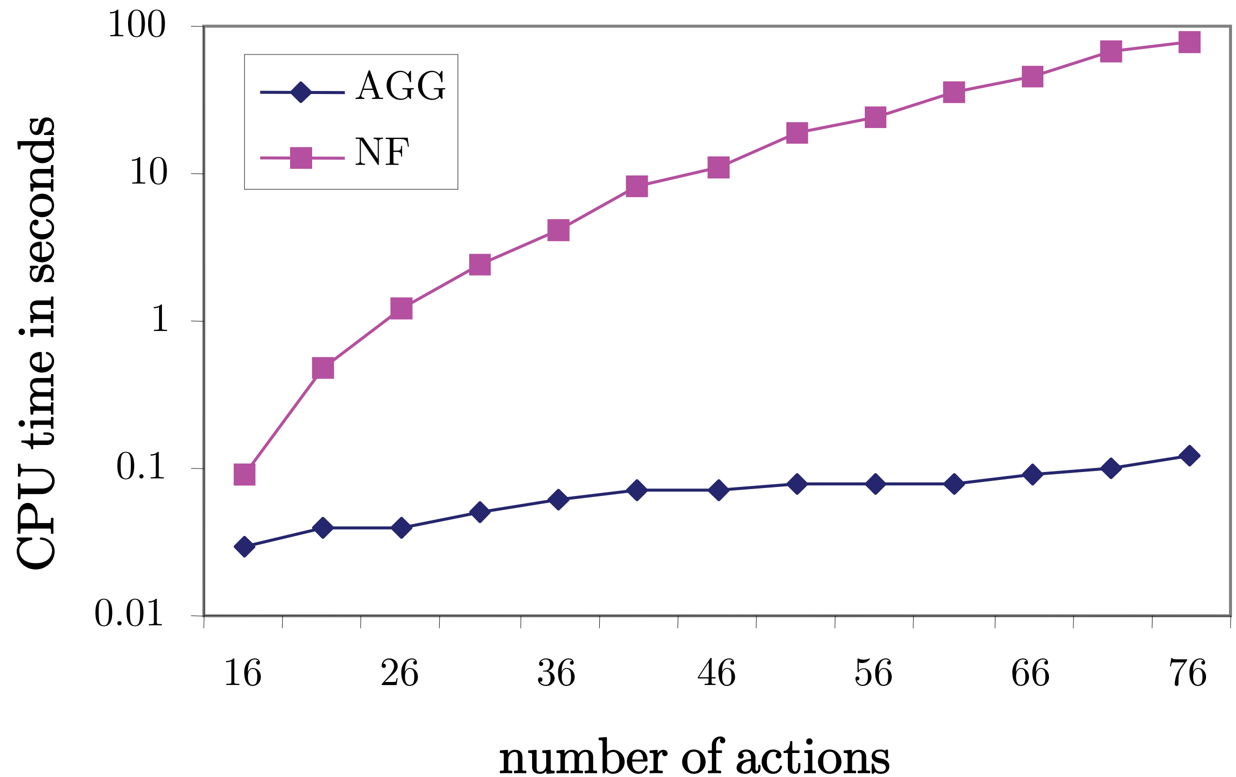
Coffee Shop Game, 5×5 grid, AGG

1000 random strategy profiles with full support

AGG grows polynomially

Experimental Results: Expected Payoff

varying ***number of actions***



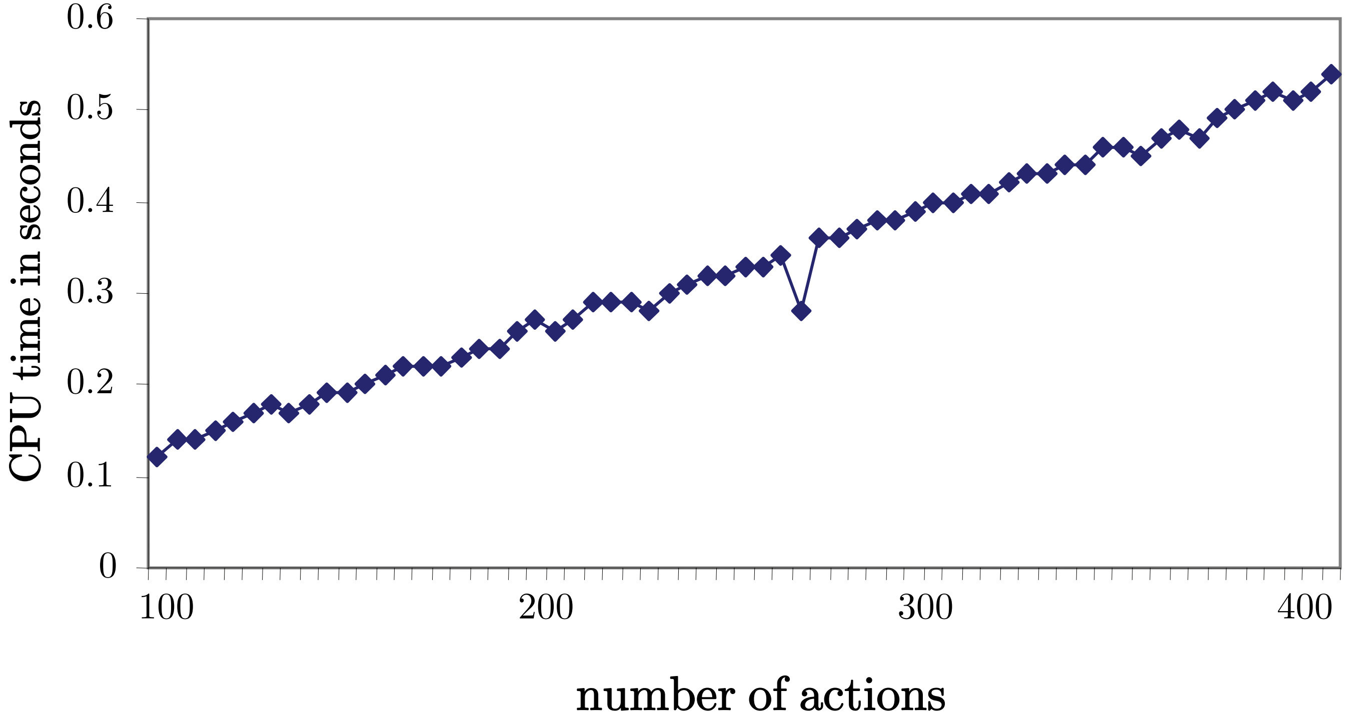
Coffee Shop Game, 4 players, ⊠ $\times$ 5 grid, AGG vs. GameTracer using

1000 random strategy profiles with full support

AGG grows linearly, NF grows as higher-order polynomial

Experimental Results: Expected Payoff

varying number of actions



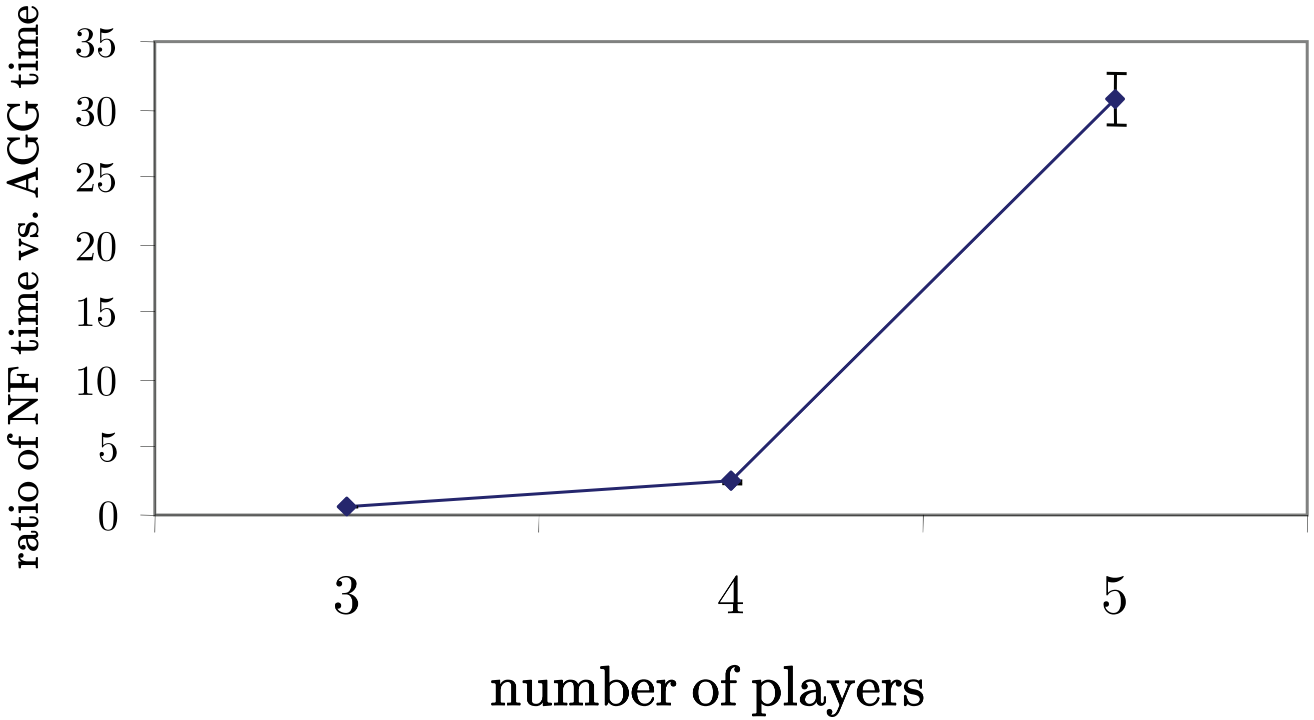
Coffee Shop Game, 4 players, ☒ × 5 grid, AGG vs. GameTracer using

1000 random strategy profiles with full support

AGG grows linearly

Experimental Results: Nash Equilibrium

varying ***number of players***



Coffee Shop Game, 4×4 grid, Govindan-Wilson Algorithm

Jacobians computed using AGGs vs. GameTracer using NF

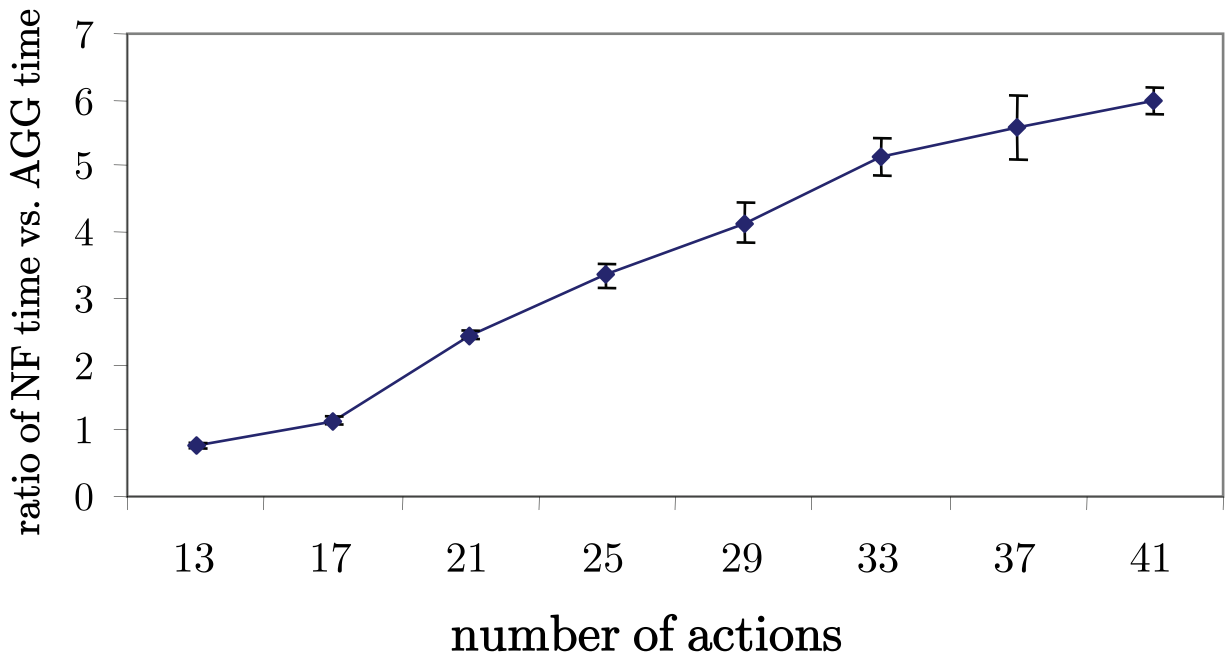
Exactly the same equilibria were found using both representations

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, ☒ × 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 - ⊠^{3} - ⍓^{2} - 0.5⌘
  ☙ ⊠, ⍓, ⌘ 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 - ⊠^{3} - ⍓^{2} - 0.5⌘
  - ⊠, ⍓, ⌘ 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 - ⊠^{3} - ⍓^{2} - 0.5⌘
  - ɞ ⊠, ⍓, ⌘ 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 - ⌧^{3} - ⍓^{2} - 0.5⌘
  ☙ ⌧, ⍓, ⌘ 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

http://www.cs.ubc.ca/~kevinlb google://"Kevin Leyton-Brown"

Job Market Game