Scaling Up Game Theory: Representation and Reasoning with Action Graph Games

Kevin Leyton-Brown
Computer Science
University of British Columbia

This talk is primarily based on papers with:

Albert Xin Jiang [AAAI 2006]

Navin A.R. Bhat [UAI 2004]

and a joint paper [GEB, to appear 2010]

and also touches on more recent joint work with
Albert Xin Jiang, David R.M. Thompson,
Avi Pfeffer, Damien Bargiacchi, and James Wright

The Kind of Games Often Studied

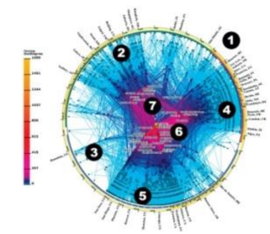
- e.g., Prisoner's Dilemma: you and an accomplice are arrested. Should you confess or stay silent?

	C	D
C	$-5, -5$	$-20, 0$
D	$0, -20$	$-1, -1$

- The analysis of such 2×2 games has proven surprisingly interesting, and has had a profound impact both on our understanding of strategic situations and on popular culture

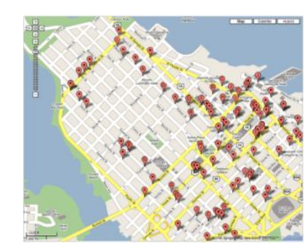
The Kind of Games We'd Like to Study

- In order to use game theory to model real systems, we need to consider games with more than two agents and two actions



- Some examples of the kinds of questions we would like to be able to answer:
  - How will heterogeneous users route their traffic in a network?
  - How will advertisers bid in a sponsored search auction?
  - Which job skills will students choose to pursue?
  - Where in a city will businesses choose to locate?

- Most GT work is analytic, not computational
- What's holding us back?
  - the size of classical game representations grows exponentially in the number of players
    - this makes all but the simplest games infeasible to write down
  - even when games can be represented, "fast" algorithms often have worst-case performance exponential in the game's size



Compact Representations

Research program for advancing the computational analysis of games:

1. find representations that can encode games of interest in **exponentially-less space** than the normal form
2. find **efficient algorithms** for working with these representations

Key representations from the literature:

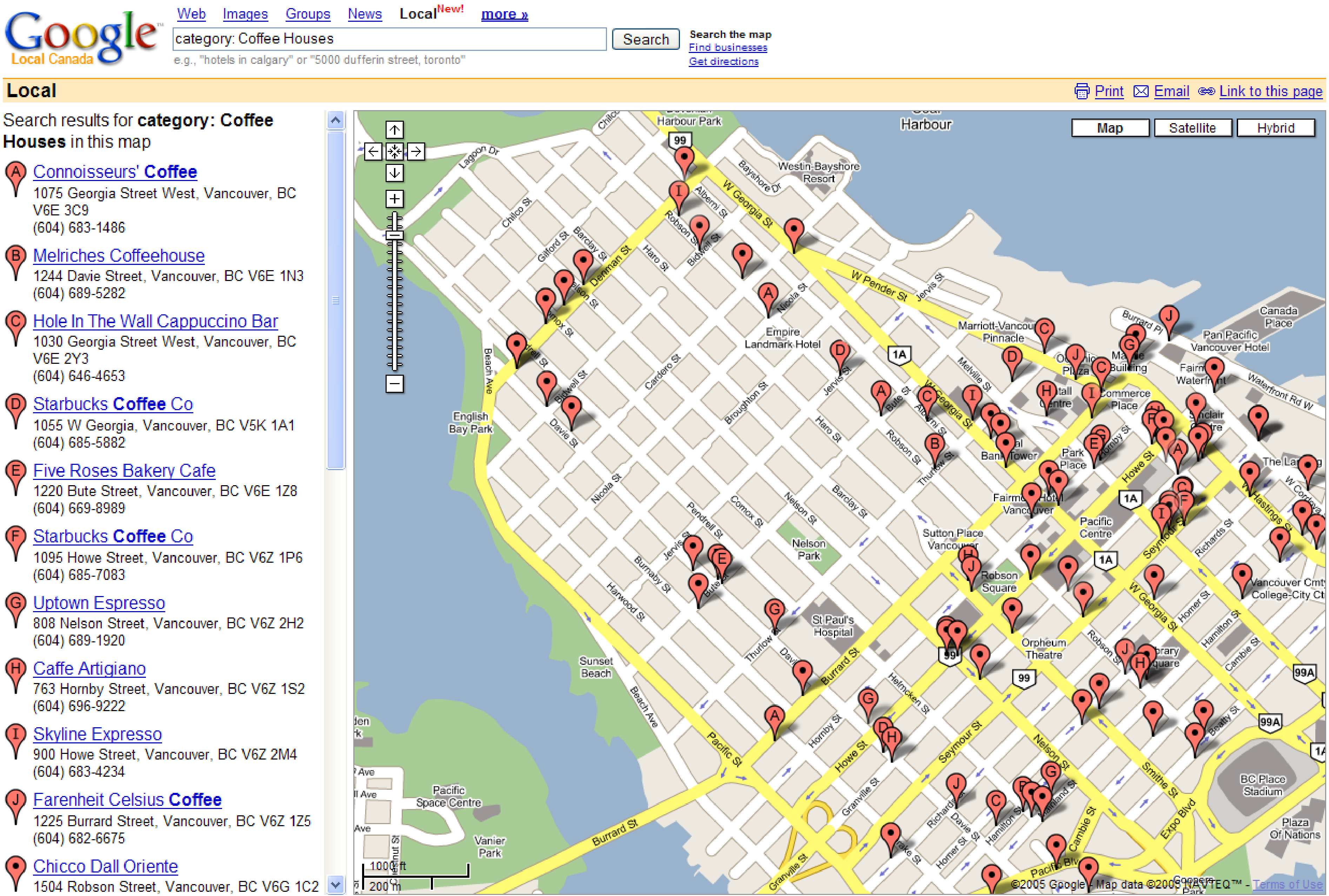
- **Graphical Games** [Kearns, Littman, Singh, 2001]
  - utility functions exhibit strict independence
    - some pairs of agents have no (direct) effect on each other's payoff
  - many efficient algorithms
  - however, none of the games discussed above are compact as GGs

- **Congestion Games** [Rosenthal, 1973; Monderer & Shapley, 1996]
  - utility functions exhibit context-specific independence
    - whether agents affect each other's payoffs can depend on the action choices they each make
  - good theoretical properties; some algorithmic results
  - however, none of the games discussed above can be represented as CGs

Overview of This Talk

1. **Basic AGGs: Definition and Examples**
2. Analyzing and Extending the Representation
3. Computing Expected Utility
4. Recent Directions

The Coffee Shop Problem



Basic 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 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 $c^{(\alpha)} \in C^{(\alpha)}$, as the elements of c that correspond to the actions $\nu(\alpha)$.

Definition 4 *A* **basic action-graph game (AGG-$\emptyset$)** *is a tuple* (N, A, G, u)*:*

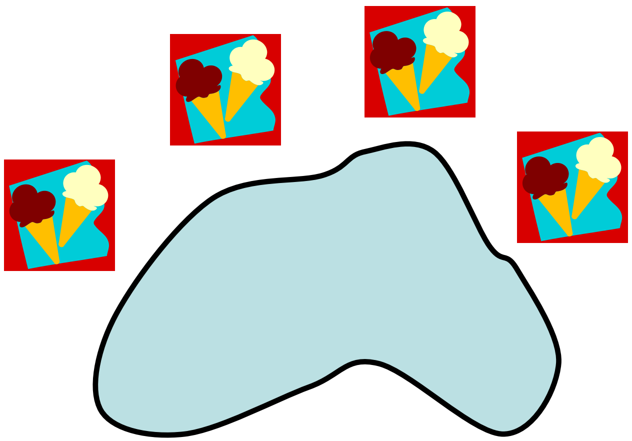
- N *is the set of agents;*
- $A = A_1 \times \cdots \times A_n$*, where* A_i *is the set of actions available to agent* i*;*
- $G = (\mathcal{A}, E)$ *is an action graph, where* $\mathcal{A} = \bigcup_{i \in N} A_i$ *is the set of distinct actions;*
- $u = (u^1, \ldots, u^{|\mathcal{A}|})$, $u^\alpha : C^{(\alpha)} \to \mathbb{R}$.

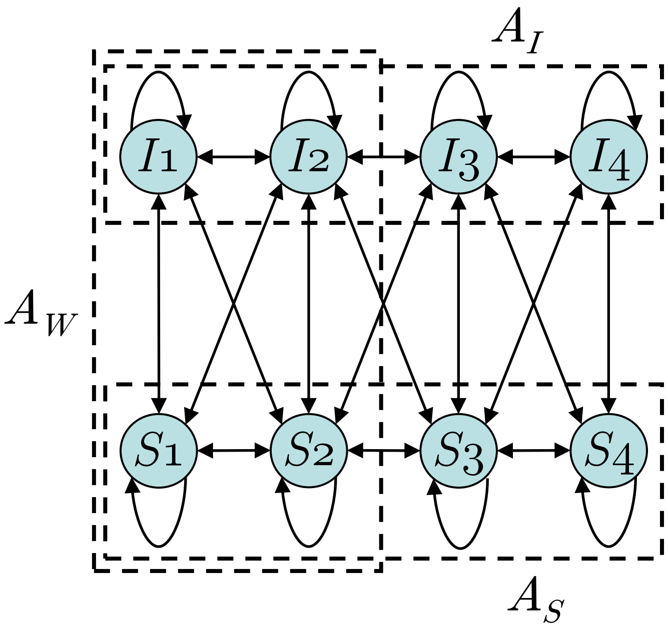
Elaborated Ice Cream Vendor Problem

Inspired by [Hotelling, 1929]

n vendors sell either ice cream or strawberries at one of four stations along a beach

- n_I ice cream (I) vendors;
- n_S strawberry (S) vendors;
- n_W can sell I/S, but only on the west side.
- competition between nearby sellers of same type; synergy between nearby different types





Notes:

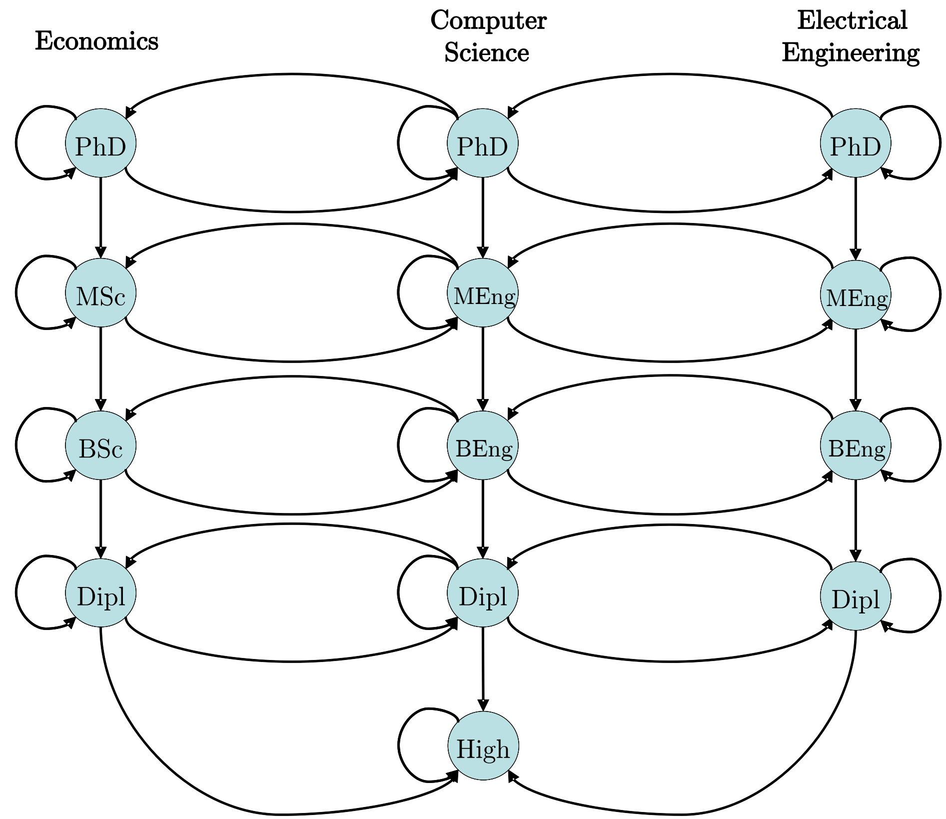
- graph structure independent of # agents
- overlapping action sets
- context-specific independence without strict independence

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 of This Talk

1. Basic AGGs: Definition and Examples
2. **Analyzing and Extending the Representation**
3. Computing Expected Utility
4. Recent Directions

Analyzing the AGG-∅ Representation

AGG-∅s can represent **any game**.

Overall, AGG-∅s 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

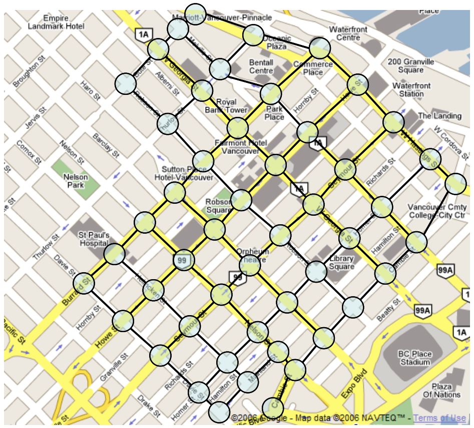
2. **Anonymity**:
   - multiple action profiles yield the same configuration

When max in-degree $\mathcal{I}$ is bounded by a constant:

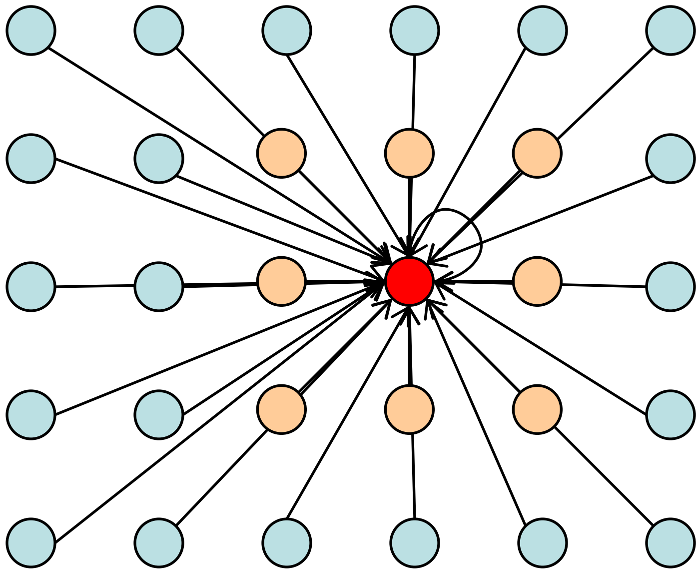
– **polynomial size:** $O(|A_{\max}|n^{\mathcal{I}})$

– in contrast, size of normal form is $\mathrm{O}(n|A_{\max}|^n)$

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: $O(|\mathcal{A}|^N)$
  - AGG-∅ representation: $O(N^{|\mathcal{A}|})$
  - when $|\mathcal{A}|$ is held constant, the AGG-∅ representation is polynomial in N
    - but still doesn't effectively capture game structure
    - given i's action, his payoff depends only on 3 quantities!

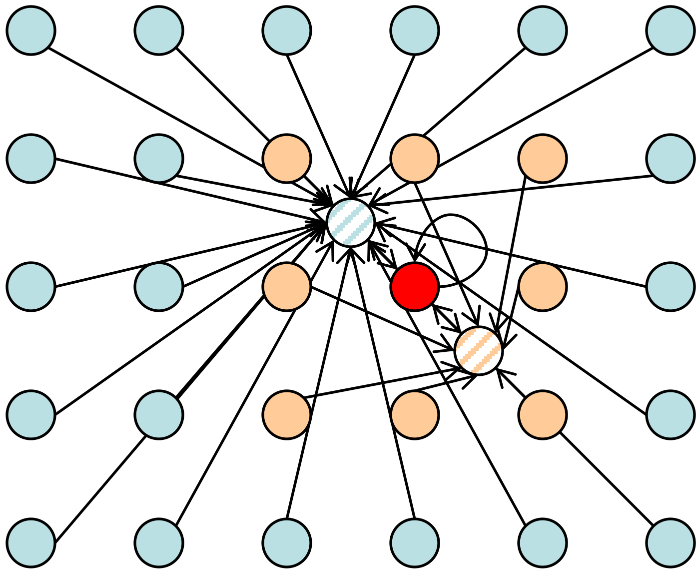


6 × 5 Coffee Shop Problem: projected action graph at the red node

AGG-FNs: Function Nodes

- To exploit this structure, introduce function nodes:
  - The "configuration" of a function node p is a (given) function of the configuration of its neighbors: $c[p] = f_p(c[\nu(p)])$

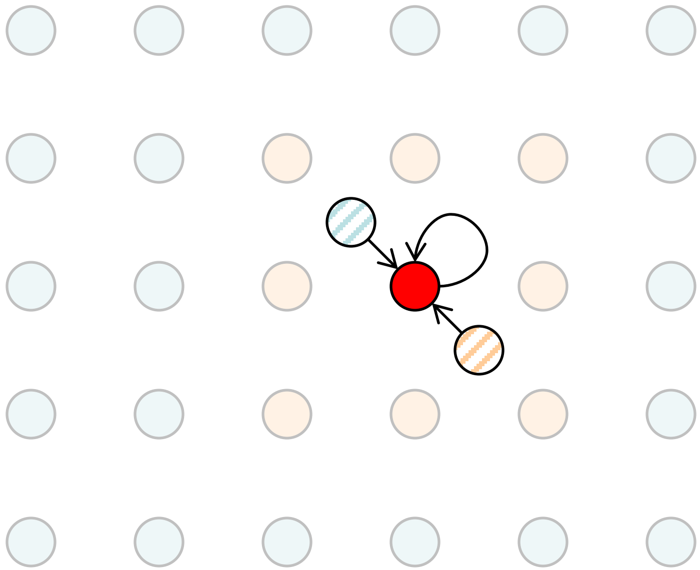
- Coffee-shop example: for each action node s, introduce:
  - a function node with adjacent actions as neighbors
    - $c[p'_s]$ = total number of shops in surrounding nodes
  - similarly, a function node with non-adjacent actions as neighbors



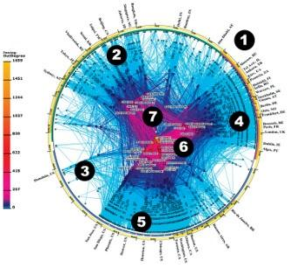
6 × 5 Coffee Shop Problem: function nodes for the red node

The Coffee Shop Problem

- Now the red node has only three incoming edges:
  - itself, the blue function node and the orange function node
  - so, the action-graph now has in-degree three
- Size of representation is now $O(N^3)$

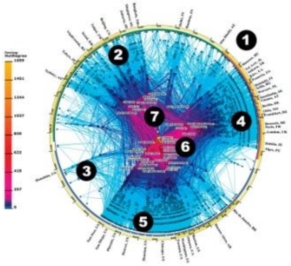


6×5 Coffee Shop Problem: projected action graph at the red node

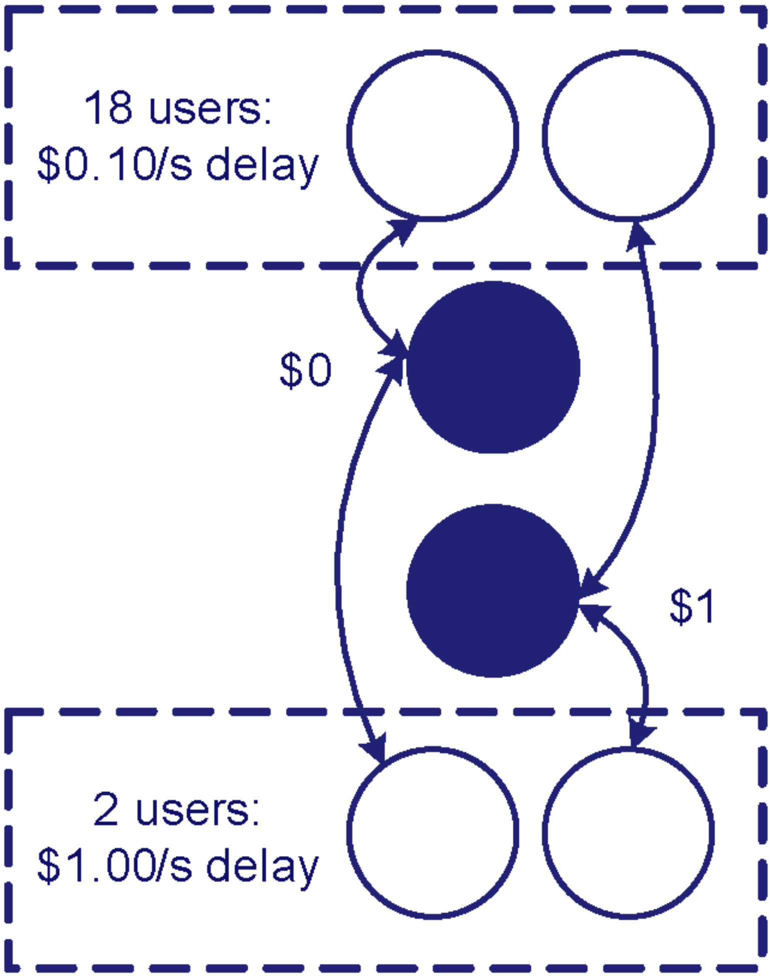


Example: Parallel Edges

Based on [Thompson, Jiang & LB, 2007]; inspired by [Odlyzko, 1998]



- Network with one source, one sink, two parallel edges
  - both edges offer identical speed
  - one is free, one costs \$1
  - latency is an additive function of the number of users on an edge
- Two classes of users
  - 18 users pay \$0.10/unit of delay
  - 2 users pay \$1.00/unit of delay
- Which edge should users choose?
- Example scales to longer paths
  - not a congestion game because of player-specific utility



Further Representational Results

- Without loss of compactness, AGGs can also encode:
  - Graphical games (AGG-$\emptyset$)
  - Symmetric games (AGG-$\emptyset$)
  - Anonymous games (AGG-FN)

- One other extension to AGGs: explicit **additive structure**
- Enables compact encoding of still other game classes:
  - Congestion games (AGG-FNA)
  - Polymatrix games (AGG-FNA)
  - Local-Effect games (AGG-FNA)

Conclusion: AGGs compactly encode **all major compact classes** of simultaneous-move games, and also **many new games** that are compact in none of these representations.

Overview of This Talk

1. Basic AGGs: Definition and Examples
2. Analyzing and Extending the Representation
3. **Computing Expected Utility**
4. Recent Directions

Computing Expected Utility

Expected utility of agent i for playing (pure) action a_i, if other agents play according to mixed-strategy profile s_{-i}:

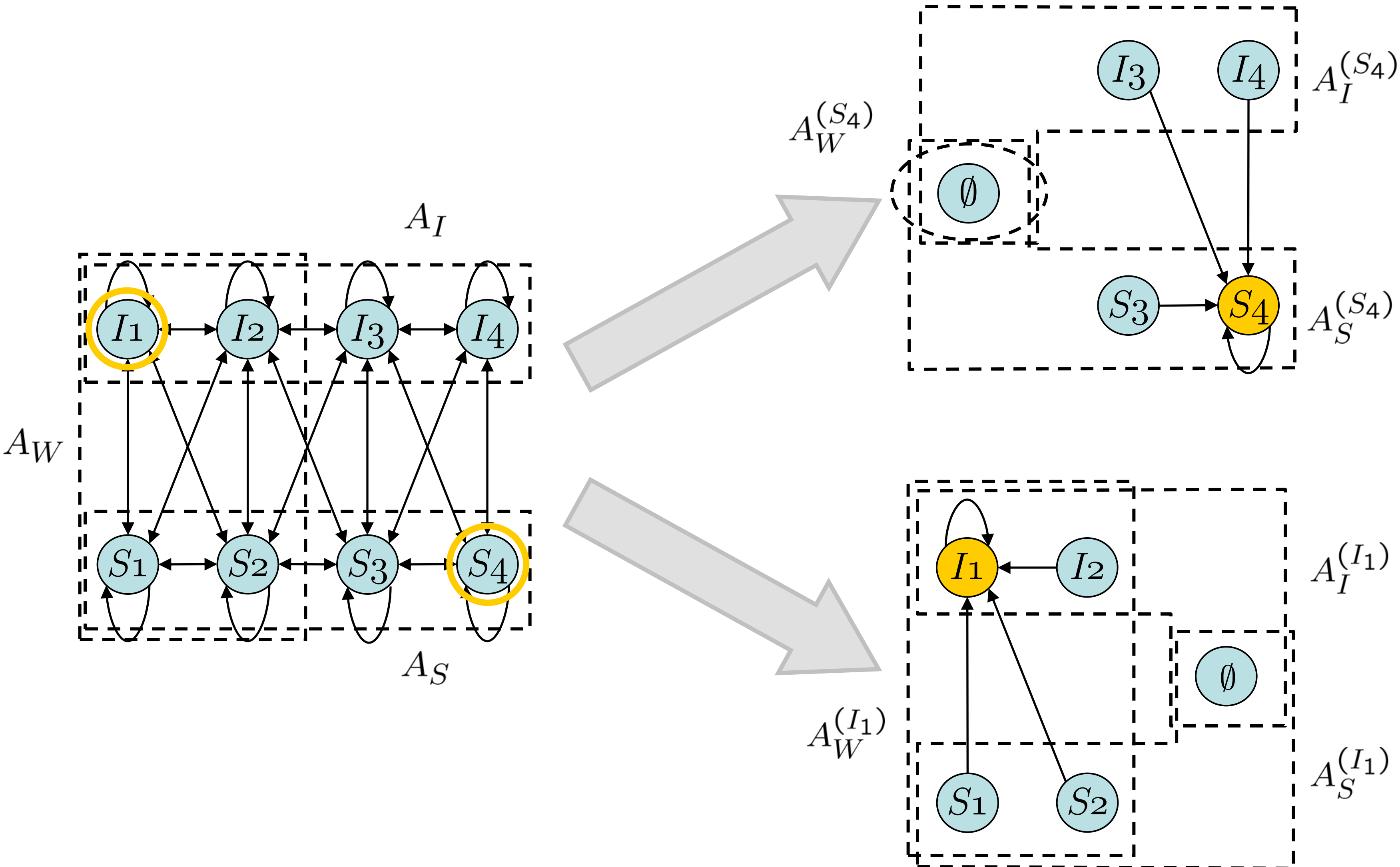
$$V^i_{a_i}(s_{-i}) \equiv \sum_{a_{-i} \in A_{-i}} u_i(a_i, a_{-i}) \, Pr(a_{-i} | s_{-i})$$

Exponential-sized set: naive algorithm is $O\left(|A_{\max}|^{n-1}\right)$

$V^i_{a_i}(s_{-i})$ is an inner-loop problem in many game-theoretic algorithms:

- **Best Response** (e.g., for multiagent reinforcement learning)
- **Govindan-Wilson** Algorithm (Nash equilibrium)
- **Simplicial Subdivision** Algorithm (Nash equilibrium)
- **Papadimitriou's** Algorithm (correlated Nash equilibrium)
- **Turocy's** Path Tracing Algorithm (quantal response equilibrium)
- **Predicted Action Distributions** under Level-k; Cognitive Hierarchy

Computing with AGG-∅s: Projection



Computing with AGG-∅s: Projection

- Projection captures context-specific independence and strict independence

$$V^i_{a_i}(s_{-i}) = \sum_{a^{(a_i)}_{-i} \in A^{(a_i)}_{-i}} u^{a_i}\left(\mathcal{C}(a_i, a^{(a_i)}_{-i})\right) Pr\left(a^{(a_i)}_{-i} | s^{(a_i)}_{-i}\right)$$

Still exponential, but smaller than before

$$Pr\left(a^{(a_i)}_{-i} | s^{(a_i)}_{-i}\right) = \prod_{j \in N \setminus \{i\}} s^{(a_i)}_j (a^{(a_i)}_j).$$

Linear-sized set

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

$\mathcal{C}(a_i, a_{-i}) \equiv$ configuration caused by a_i, a_{-i}

$\mathcal{S}\,(c) \equiv$ set of pure action profiles giving rise to c

Computing with AGG-∅s: Anonymity

- Writing in terms of the configuration captures anonymity

$$V^i_{a_i}(s_{-i}) = \sum_{c^{(a_i)}_{-i} \in C^{(a_i)}_{-i}} u^{a_i}\left(\mathcal{C}\left(a_i, c^{(a_i)}_{-i}\right)\right) Pr\left(c^{(a_i)}_{-i} | s^{(a_i)}_{-i}\right)$$

Polynomial-sized set

$$Pr\left(c^{(a_i)}_{-i} | s^{(a_i)}_{-i}\right) = \sum_{a^{(a_i)}_{-i} \in \mathcal{S}\left(c^{(a_i)}_{-i}\right)} Pr\left(a^{(a_i)}_{-i} | s^{(a_i)}_{-i}\right)$$

Exponential-sized set

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

$\mathcal{C}(a_i, c_{-i}) \equiv$ configuration caused by a_i, c_{-i}

$\mathcal{S}(c) \equiv$ set of pure action profiles giving rise to c

Dynamic Programming

- Can we **do better** computing $Pr\left(c_{-i}^{(a_i)} | s_{-i}^{(a_i)}\right)$? Note that
  - the players' mixed strategies are independent
    - s is a product probability distribution
  - each player affects a configuration c independently

- We can use **dynamic programming** to compute the probability of a configuration:
  - base case: zero agents and the mixed strategy s_0:
    - $C_0 = \{c_0\}$
    - $c_0 = [0, \ldots, 0]$
    - $P_0(c_0) = 1$
  - then add agents **one by one**:
    - C_k: the set of configurations that can be built by adding any action from the support of player k's mixed strategy to any configuration from $C_{k\text{-}1}$
    - $P_k(c_k) = \sum_{\substack{(c_{k-1}, a_k), \\ \mathcal{C}(c_{k-1}, a_k) = c_k}} s_k(a_k) \cdot P_{k-1}(c_{k-1})$

Computing with AGGs: Complexity

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

- Complexity of our approach:
  $$O\left(n^{\mathcal{I}} poly(n) poly\left(|A_{\max}|\right)\right)$$
- Exponential speedup vs. standard approach:
  $$O\left(|A_{\max}|^{n-1} poly(n) poly(|A_{\max}|)\right)$$

In AGG-FNs, players are no longer guaranteed to affect c independently

- but the DP algorithm still works when function nodes can be expressed using some commutative, associative operator

Computing Expected Utility

$V^i_{a_i}(s_{-i})$ is an inner-loop problem in many game-theoretic algorithms:

- Best Response (e.g., for multiagent reinforcement learning)
- Govindan-Wilson Algorithm (Nash equilibrium)
- Simplicial Subdivision Algorithm (Nash equilibrium)
- Papadimitriou's Algorithm (correlated Nash equilibrium)
- Turocy's Path Tracing Algorithm (quantal response equilibrium)
- Predicted Action Distributions under Level-k; Cognitive Hierarchy

Because we compute $V^i_{a_i}(s_{-i})$ exactly, our expected utility algorithm yields an exponential speedup in every one of these algorithms, whenever the AGG is exponentially smaller than the normal form.

Overview of This Talk

1. Basic AGGs: Definition and Examples
2. Analyzing and Extending the Representation
3. Computing Expected Utility
4. **Recent Directions**
    1. computing pure strategy equilibria
    2. analyzing sponsored search auctions
    3. temporal AGGs
    4. Bayesian AGGs
    5. free software tools

(1) Computing Pure-Strategy Equilibrium

- Pure Nash equilibrium is often a more interesting solution concept than mixed Nash equilibrium

- It also presents a very computationally different problem
  - PSNE in normal form admits a very simple polytime algorithm
    - just check every action profile
  - For AGG-$\emptyset$s the representation can be exponentially smaller
    - thus, the same algorithm is exponential time

Theorem (Conitzer, personal communication; also proven independently in (Daskalakis et al. 2008)): The problem of determining whether a pure Nash equilibrium exists in an AGG-$\emptyset$ is NP-complete, even when the AGG-$\emptyset$ is symmetric and has max in-degree of three.

(1) Computing PSNEs in AGG-∅s

[Jiang & LB, 2007]

We propose a **message passing algorithm**:

- partition action graph into subgraphs (via tree decomposition)
- construct equilibria of the game from equilibria of games played on subgraphs

This algorithm finds PSNE in polynomial time for every **symmetric AGG-∅ that has bounded treewidth.**

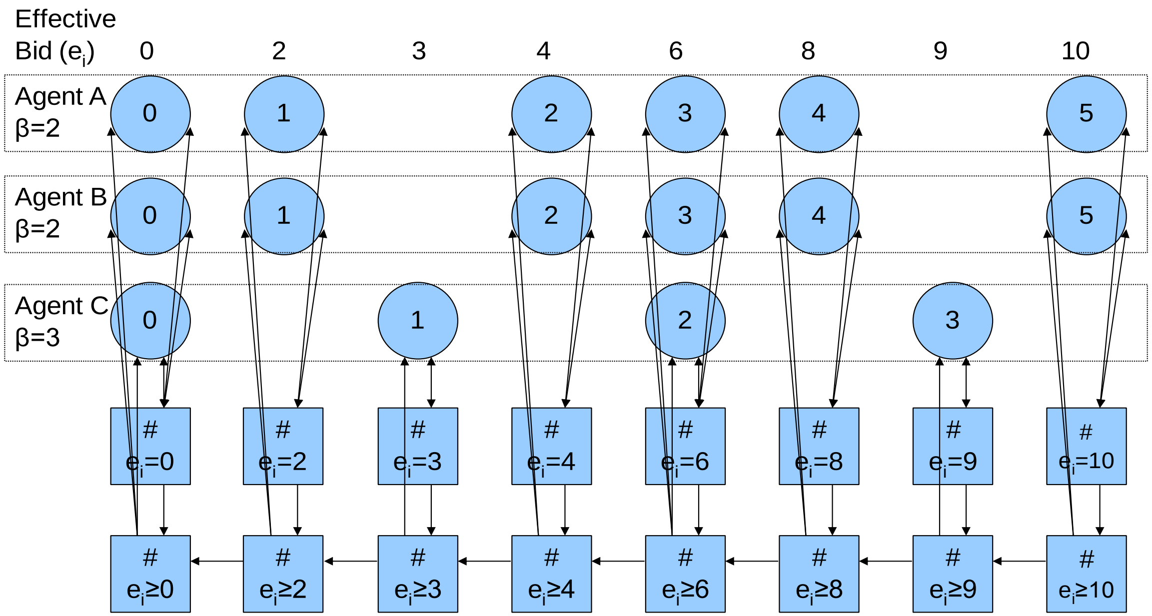
- it can also be applied to other bounded-treewidth settings

- Generalizes earlier algorithms
  - finding pure equilibria in **graphical games**
    [Gottlob, Greco, & Scarcello 2003; Daskalakis & Papadimitriou 2006]
  - finding pure equilibria in **simple congestion games**
    [Ieong, McGrew, Nudelman, Shoham, & Sun 2005]

(2) Sponsored Search Auctions

[Thompson & LB, 2008; 2009]

- Position auctions are used to sell $10Bs of keyword ads
- Some theoretical analysis, but based on strong assumptions
  - Unknown how different auctions compare in more general settings
- Idea: analyze the auctions computationally
  - Main hurdle: ad auction games are large; infeasible as normal form

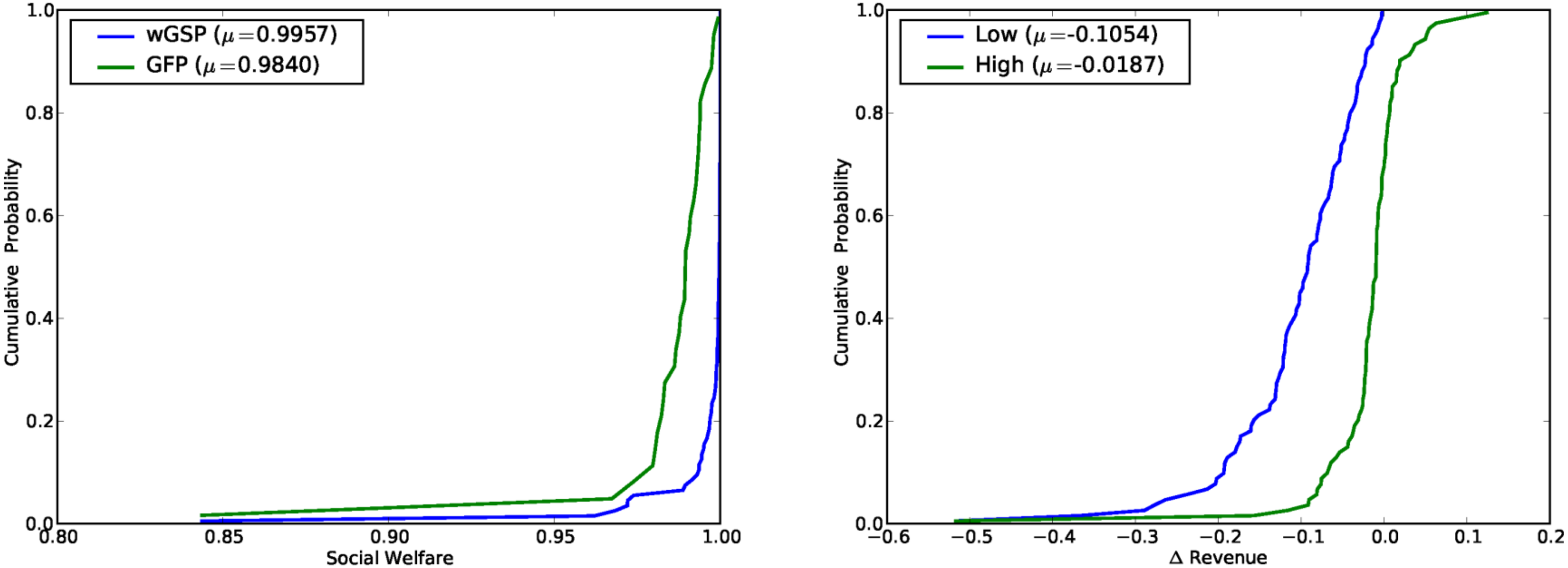


AGG-FN representation of a Weighted, Generalized First-Price (GFP) Auction

(2) Sponsored Search Auctions

[Thompson & LB, 2008; 2009]

- Position auctions are used to sell \$10Bs of keyword ads
- Some theoretical analysis, but **based on strong assumptions**
  - Unknown how different auctions compare in more general settings
- Idea: **analyze the auctions computationally**
  - Main hurdle: ad auction games are large; infeasible as normal form



Social welfare and revenue of EOS auction model

(3) Temporal Action Graph Games

[Jiang, LB & Pfeffer, 2009]

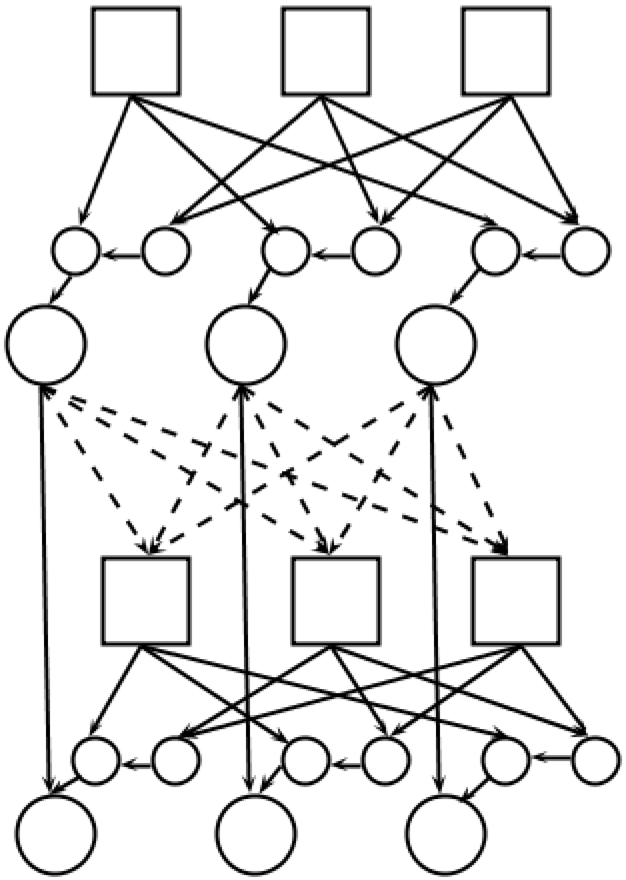
Goal: extend AGGs to temporal settings

- Model: An AGG-FN played over a series of discrete time steps
  - at each time step, a subset of players move
  - action counts on the action nodes grow over time
- Allow payoff uncertainty using random variables that are realized at a given time step
- Imperfect information: players may condition their actions on a given set of observed previous actions, chance variables and action counts
- Utility functions: action-specific and time-specific

(3) Properties of TAGGs

[Jiang, LB & Pfeffer, 2009]

- Can compactly represent a wide range of dynamic games, including:
  - arbitrary MAIDs [Milch & Koller, 2001]
  - games whose straightforward MAID representations are not compact
- Can be efficiently encoded as MAIDs by introducing deterministic chance nodes
- Efficient computation of expected utility
  - exploit anonymity and context-specific independence as in AGG-∅s
  - also exploit the temporal structure
  - as with AGG-∅s, can be leveraged to yield exponential speedups in computation (Nash equilibrium, etc.)



(4) Bayesian Games

- TAGGs aren't the most appropriate way of representing simultaneous-move Bayesian games
  - indeed, while such models are widely used (e.g., in auction theory), the setting has largely been neglected by the computational game theory community
- As far as we know, there are no representations or algorithms targeting general BNE computation
- This leaves two general approaches, both of which make use of complete-information Nash algorithms:
  1. Induced normal form
     - one action for each pure strategy (mapping from type to action)
     - set of players unchanged
  2. Agent form
     - one player for each type of each of the BG's players
     - action space unchanged

(4) Bayesian AGGs

[Jiang & LB, work under review 2010]

Bayesian AGG: an AGG-like representation of a Bayesian game's utility functions, which compactly encodes its agent form:

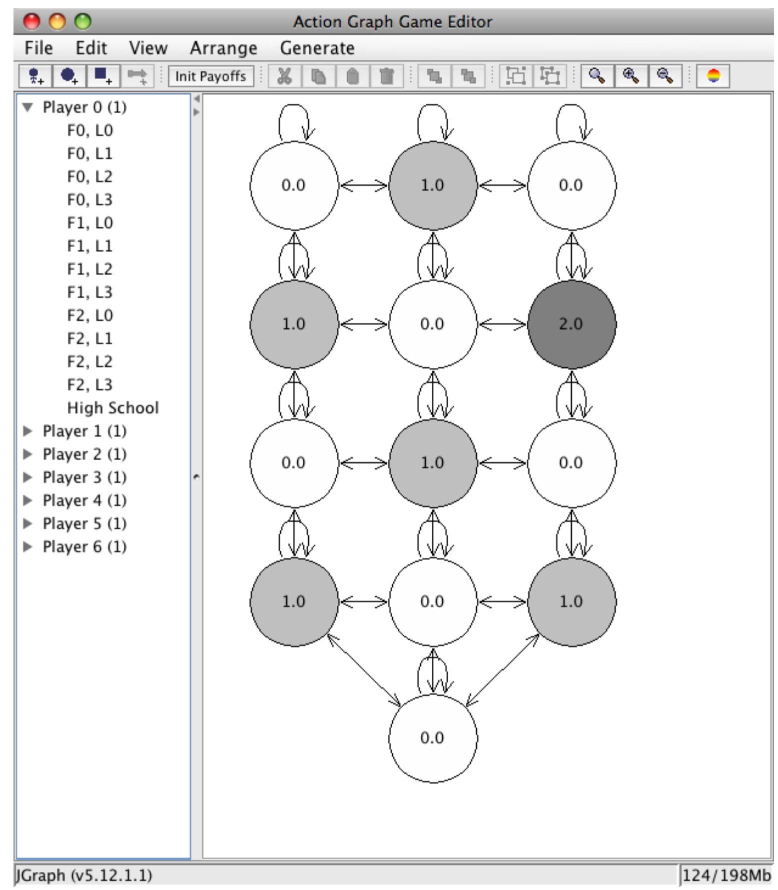
- Bayesian network for the joint type distribution
- A (potentially separate) action graph for each type of each agent
- A utility function that depends on which types are realized and on the actions taken by the other agents of the appropriate types

- Representation size grows polynomially in $|\Theta|$, $|A|$, n, when action graph has constant-bounded in-degree
  - Exponential savings over an unstructured Bayesian game
- When types are independent, expected utility can be computed in time polynomial in the size of the BAGG
- When types are not independent, expected utility can still be computed in polynomial time when an induced Bayesian network has bounded treewidth.

(5) Free Software Tools

[Jiang, Bargiacchi & LB, 2007–2010]

- Goal: make it easier for other researchers to use AGGs
- Equilibrium computation algorithms:
  - Govindan-Wilson (NE)
  - Simplicial Subdivision (NE)
  - Papadimitriou (CE) ** in progress*
  - Turocy (QRE) ** in progress*
- GAMUT:
  - extended to support AGGs
- Action Graph Game Editor:
  - creates AGGs graphically
  - facilitates entry of utility fns
  - supports "player classes"
  - auto creates game generators
  - visualizes eq. on the action graph



Conclusions

- **AGGs compactly represent games** exhibiting context-specific independence, anonymity and/or additive structure
- **Generalizes all major, existing compact representations** of simultaneous-move games
  - graphical games, congestion games, many others
- Recent directions:
  - Polytime algorithm for computing **pure strategy Nash equilibrium** (bounded treewidth; symmetric AGG-∅)
  - modeling and comparing **sponsored search auctions**
  - extending AGGs to **temporal settings**
  - extending AGGs to **Bayesian games**
  - developing **free software tools**

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