Computing Pure Nash Equilibria in Symmetric Action Graph Games

Albert Xin Jiang Kevin Leyton-Brown
Department of Computer Science
University of British Columbia
{jiang;kevinlb}@cs.ubc.ca

July 26, 2007

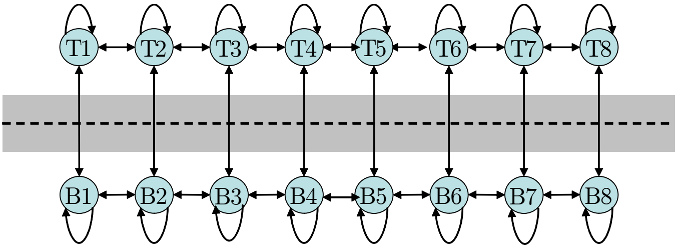
Outline

Outline

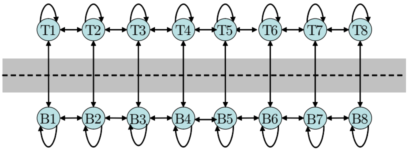
1. Action Graph Games
2. Pure Nash Equilibria
3. Computing Pure Equilibira in Symmetric AGGs
4. Algorithm
5. Conclusions & Future Work

Example: Location Game

- each of n agents wants to open a business
- actions: choosing locations
- utility: depends on
  - the location chosen
  - number of agents choosing the same location
  - numbers of agents choosing each of the adjacent locations



Game on a graph



- This can be modeled as a game played on a directed graph:
  - each player has one token to put on one of the nodes;
  - utility depends on:
    - the node chosen
    - configuration of tokens over neighboring nodes
- Action Graph Games (Bhat & Leyton-Brown 2004, Jiang & Leyton-Brown 2006)
  - fully expressive, compact representation of games
  - exploits anonymity, context specific independence

Definitions

Definition (action graph)

An action graph is a tuple (S, E), where S is a set of nodes corresponding to *distinct actions* and E is a set of directed edges.

- Each agent i's set of available actions: $S_i \subseteq S$
- Neighborhood of node s: $\nu(s) \equiv \{s' \in S | (s', s) \in E\}$

Definitions

Definition (action graph)

An action graph is a tuple (S, E), where S is a set of nodes corresponding to *distinct actions* and E is a set of directed edges.

- Each agent i's set of available actions: $S_i \subseteq S$
- Neighborhood of node s: $\nu(s) \equiv \{s' \in S | (s', s) \in E\}$

Definition (configuration)

A configuration D is an $|S|$-tuple of integers $(D[s])_{s \in S}$. $D[s]$ is the number of agents who chose the action $s \in S$. For a subset of actions $X \subset S$, let $D[X]$ denote the restriction of D to X. Let $\Delta[X]$ denote the set of restricted configurations over X.

Action Graph Games

Definition (action graph game)

An action graph game (AGG) is a tuple $\langle N, (S_i)_{i\in N}, G, u\rangle$ where

- N is the set of agents
- S_i is agent i's set of actions
- $G = (S, E)$ is the action graph, where $S = \bigcup_{i\in N} S_i$ is the set of distinct actions
- $u = (u^s)_{s\in S}$, where $u^s : \Delta[\nu(s)] \mapsto \mathbb{R}$

Action Graph Games

Definition (action graph game)

An action graph game (AGG) is a tuple $\langle N, (S_i)_{i \in N}, G, u \rangle$ where

- N is the set of agents
- S_i is agent i's set of actions
- $G = (S, E)$ is the action graph, where $S = \bigcup_{i \in N} S_i$ is the set of distinct actions
- $u = (u^s)_{s \in S}$, where $u^s : \Delta[\nu(s)] \mapsto \mathbb{R}$

Definition (symmetric AGG)

An AGG is symmetric if all players have identical action sets, i.e. if $S_i = S$ for all i.

AGG Properties

- AGGs are fully expressive
- Symmetric AGGs can represent arbitrary symmetric games
- Representation size $||\Gamma||$ is polynomial if the in-degree $\mathcal{I}$ of G is bounded by a constant
- Any graphical game (Kearns, Littman & Singh 2001) can be encoded as an AGG of the same space complexity.
- AGG can be exponentially smaller than the equivalent graphical game & normal form representations.

Outline

Pure Nash Equilibria

Action profile: $\mathbf{s} = (s_1, \ldots, s_n)$

Definition (pure Nash equilibrium)

An action profile s is a *pure Nash equilibrium* of the game Γ if for all $i \in N$, s_i is a best response to s_{-i} (i.e. for all $s_i' \in S_i$, $u_i(s_i, s_{-i}) \geq u_i(s_i', s_{-i})$).

- not guaranteed to exist
- often more interesting than mixed Nash equilibria

Complexity of Finding Pure Equilibria

Checking every action profile:

- linear time in normal form size
- worst-case exponential time in AGG size

Complexity of Finding Pure Equilibria

Checking every action profile:

- linear time in normal form size
- worst-case exponential time in AGG size

We focus on symmetric AGGs

- only need to consider configurations

Theorem (Conitzer, personal communication)

The problem of determining whether a pure Nash equilibrium exists in a symmetric AGG is NP-complete, even when the in-degree of the action graph is at most 3.

Complexity of Finding Pure Equilibria

Checking every action profile:

- linear time in normal form size
- worst-case exponential time in AGG size

We focus on symmetric AGGs

- only need to consider configurations

Theorem (Conitzer, personal communication)

The problem of determining whether a pure Nash equilibrium exists in a symmetric AGG is NP-complete, even when the in-degree of the action graph is at most 3.

For symmetric AGGs with bounded $|S|$:

- number of configurations is polynomial
- pure equilibria can be found in poly time by enumerating configurations

Main Results

Our dynamic programming approach:

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

Tractable class: symmetric, bounded treewidth and in-degree[1].

- our approach can be extended beyond symmetric AGGs

[1] different from published version of paper

Main Results

Our dynamic programming approach:

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

Tractable class: symmetric, bounded treewidth and in-degree[1].

- our approach can be extended beyond symmetric AGGs

Related Work:

- (Gottlob, Greco, & Scarcello 2003) and (Daskalakis & Papadimitriou 2006)
  - finding pure equilibria in graphical games
- (Ieong, McGrew, Nudelman, Shoham, & Sun 2005)
  - finding pure equilibria in singleton congestion games
  - can be represented as AGGs with only self edges

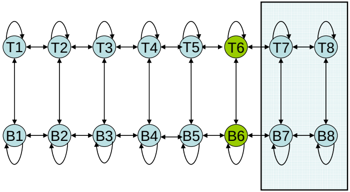
[1] different from published version of paper

Outline

1. Action Graph Games
2. Pure Nash Equilibria
3. Computing Pure Equilibira in Symmetric AGGs
4. Algorithm
5. Conclusions & Future Work

Restricted Game

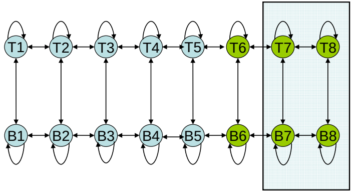
- game played by a subset of players: $n' \leq n$
- actions restricted to $R \subset S$
- utility functions same as in original AGG
  - need to specify configuration of neighboring nodes not in R



- *restricted game* $\Gamma(n', R, D[\nu(R)])$

Partial Solution

- want to use equilibria of restricted games as building blocks
- also need $D[\nu(X)]$ to specify the restricted game



Definition (partial solution)

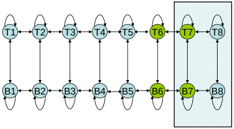
A *partial solution* on $X \subseteq S$ is a configuration $D[X \cup \nu(X)]$ such that $D[X]$ is a pure equilibrium of the restricted game $\Gamma(\#D[X], X, D[\nu(X)])$.

A partial solution describes a restricted game as well as a pure equilibrium of it.

Extending partial solutions

- Problem: combining two partial solutions on two non-overlapping restricted games does not necessarily produce an equilibrium of the combined game
  - configurations may be inconsistent, or
  - player might profitably deviate from playing in one restricted game to another
- keeping all partial solutions: impractical as sizes of restricted games grow
- we would like sufficient statistics that summarize partial solutions

Sufficient statistic



Sufficient Statistic: a tuple consisting of

1. configuration over
   - outside neighbours: $\nu(X)$
   - inside nodes that are neighbors of outside nodes: $\nu(\overline{X})$
2. $\#$ of agents playing in X
3. utility of the worst-off player in X.
4. best utility an outside player can get by playing in X.
   - different cases for deviation from $\nu(X)$

Number of distinct tuples:

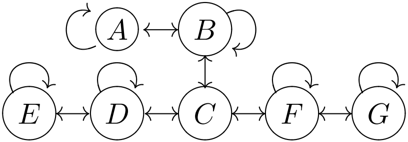
- polynomial for action graphs of bounded treewidth and in-degree[2]

[2] different from published version of paper

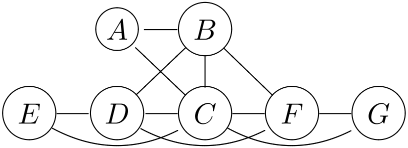
Outline

Example: an action graph and its primal graph

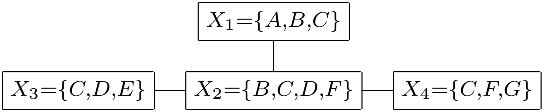
- action graph:



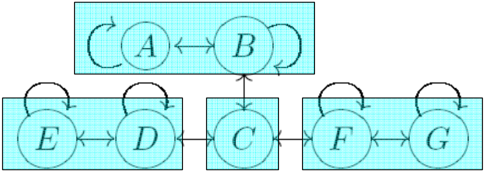
- primal graph: make each neighborhood a clique



Example: tree decomposition

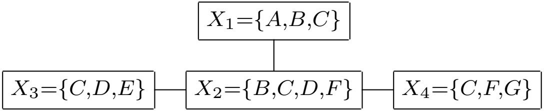


This corresponds to the following partition:

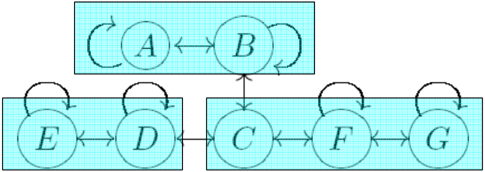


Example: combining restricted games

combine restricted games in bottom-up order: from leaves to root

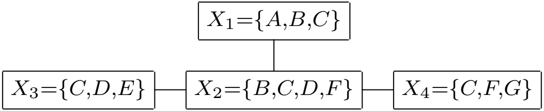


partition after combining {C} and {F,G}:

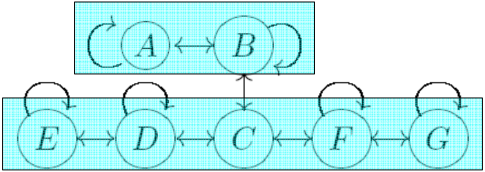


Example: combining restricted games

combine restricted games in bottom-up order: from leaves to root



partition after combining {D,E} and {C,F,G}:



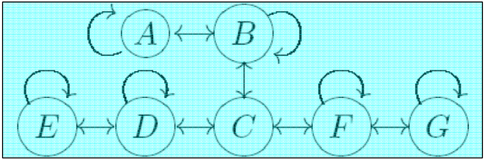
Example: combining restricted games

combine restricted games in bottom-up order: from leaves to root

$X_1 = \{A, B, C\}$

$X_3 = \{C, D, E\}$ — $X_2 = \{B, C, D, F\}$ — $X_4 = \{C, F, G\}$

partition after combining {A,B} and {C,D,E,F,G}:



Summary of Algorithm

- given a symmetric AGG with treewidth w:
    - construct tree decomposition of width w
        - poly time if w bounded by a constant
    - construct tree decomposition of primal graph with width at most $(w+1)\mathcal{I}-1$
    - combine restricted games in bottom-up order: from leaves to the root

Summary of Algorithm

- given a symmetric AGG with treewidth w:
  - construct tree decomposition of width w
    - poly time if w bounded by a constant
  - construct tree decomposition of primal graph with width at most $(w+1)\mathcal{I}-1$
  - combine restricted games in bottom-up order: from leaves to the root

Theorem

For symmetric AGGs with bounded treewidth and in-degree[a], our algorithm determines the existence of pure Nash equilibria in polynomial time.

[a] different from published version of paper

- then a top-down pass computes the equilibria

Outline

Conclusions & Future Work

- dynamic programming approach for computing pure equilibria in AGGs
- poly-time algorithm for symmetric AGGs with bounded treewidth and in-degree
- our approach can be extended to general AGGs
  - different set of sufficient statistics
  - related algorithms for graphical games (Daskalakis & Papadimitriou 2006) and singleton congestion games (Ieong et al 2005) become special cases of our approach