Computational Game Theory and Its Applications

Hau Chan
University of Nebraska-Lincoln

Summer School on Game Theory and Social Choice
June 21 - June 24 and July 9 - July 13, 2021
Game theory has numerous applications!

Q: What is game theory? How can we use it to model these real-world scenarios?
We Don't Live In A Vacuum!

• We interact with other agents in our environment to make rational decisions.

• Game theory is a mathematical tool that allows us to reason about the strategic interaction of self-interest and rational agents in a given environment.

• The construct provides a set of framework that characterizes the rational outcomes in such environment of strategic agents.
Roadmap

1. Intro
2. Classical Game Theory
3. Computational Game Theory
4. Application of Game Theory
Decision Problem

• Single-Agent Decision Problem
  – An individual or player or agent faces a situation in which he has to choose one of several alternatives/actions
  – Each action will result in some outcome
  – The player will bear the consequences of that outcome (i.e., receive some value or utility)

Should I gamble?

<table>
<thead>
<tr>
<th>Yes</th>
<th>Utility: 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Utility: -5</td>
</tr>
</tbody>
</table>

• Game Theory = Multiple-Agent Decision Problem
  – Your actions + other agents’ actions = outcomes
  – How should they act strategically and rationally?
People **Strategically** Interact With Each Other Everyday!

- **Rock-Paper-Scissors** (2 players)
- **Penalty Kicks** (2 players)
- **Road Congestions** (Multiple players/drivers)

**Q: What are the behaviors of the players?**

**A: Game Theory**
Example Games: “Chicken”

- 2 players/cars driving towards each other
- They can either go straight (S) or go around (A)
- If one player goes straight, that player wins ($$$$$$?)
- If both go straight, they both hurt

\[
\begin{array}{c|cc}
& A & S \\
\hline
A & 0, 0 & -1, 1 \\
S & 1, -1 & -5, -5 \\
\end{array}
\]
Example Games:
“A Toy Security Game”

<table>
<thead>
<tr>
<th>Attacker</th>
<th>-1, 1</th>
<th>1, -1</th>
<th>1, -1</th>
<th>0, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airline</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Games as Mathematical Abstractions

**A toy security game**

<table>
<thead>
<tr>
<th></th>
<th>-1, 1</th>
<th>1, -1</th>
<th>1, -1</th>
<th>0, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attacker</strong></td>
<td>![Attacker Image]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Airline</strong></td>
<td>![Airline Image]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**In general**

**Game:**

1. A set of $n$ agents/players
2. For each agent $i$, there is a set of actions (pure-strategies) $S_i$
3. For each agent $i$, there is a utility function $u_i : S_1 \times \ldots \times S_n \rightarrow \mathbb{R}$
Games

Solution Concept [Nash, 1950]

Given a game, we want to use solution/equilibrium concept(s) to rationalize agents’ behaviors.

Nash Equilibrium (NE) ➔ Intuition: Each agent acts to maximize the (expected) utility given the actions of other agents.

Let $A(S_i)$ be the set of prob. distr. over $S_i$. $(x_i, x_{-i}) \in A(S_i) \times \ldots \times A(S_n)$ is a mixed-strategy NE iff, for all $i$, for all $x_i \in A(S_i)$, $u_i(x_i, x_{-i}) \geq u_i(x_i, x_{-i})$.

We can show that there is a (mixed-strategy) Nash Equilibrium:

- **Attacker** attacks with prob. $1/3$ (utility $= 1/3$)
- **Airline** invests with prob. $1/3$ (utility $= -1/3$)
Normal-form Games

1. A set of n agents/players

2. For each agent i, there is a set of actions (pure-strategies) \( S_i \)

3. For each agent i, there is a utility function \( u_i : S_1 \times \cdots \times S_n \rightarrow \mathbb{R} \)

<table>
<thead>
<tr>
<th></th>
<th>2, 2</th>
<th>-1, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7, -8</td>
<td>0, 0</td>
<td></td>
</tr>
</tbody>
</table>

Static Games of Complete Information:
• All players know the game they are playing
• One-shot simultaneous settings where each player is choosing a strategy without knowing the choices of the other players

Solution Concepts:
• Pure-strategy Nash Equilibrium or Mixed-strategy Nash Equilibrium
Other Types or Representations of Games*

- **Normal-form games**

<table>
<thead>
<tr>
<th></th>
<th>2, 2</th>
<th>-1, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7, -8</td>
<td>0, 0</td>
<td></td>
</tr>
</tbody>
</table>

- **Stochastic games**

  - 1, 1 1, 0
  - 0, 1 0, 0

- **Bayesian games**

  - Row player type 1 (prob. 0.5)
    - U: 4 6
    - D: 2 4
  - Column player type 1 (prob. 0.5)
    - U: 4 6
    - D: 4 6
  - Row player type 2 (prob. 0.5)
    - U: 2 4
    - D: 4 2
  - Column player type 2 (prob. 0.5)
    - U: 2 2
    - D: 4 2

- **Stackelberg games (commitment)**

<table>
<thead>
<tr>
<th></th>
<th>2, 1</th>
<th>4, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 0</td>
<td>3, 1</td>
<td></td>
</tr>
</tbody>
</table>

*Modified Slides from Vincent Conitzer
Extensive-form Games

• One of normal-form games’ drawbacks is its inability to capture games that unfold over time.

  • There is no way to represent situations in which the order of moves might be important.

• Extensive-form games allow us to represent formally such sequential strategic situations and apply strategic reasoning to these new representations.
Extensive-form Games

1. A set of n agents/players

2. A rooted tree (game tree)
   - Each leave stores agents’ utilities
   - Other nodes are partitioned to n+1 partitions (i.e., P₁, P₂, ..., Pₙ₊₁)
     - Each node in Pₙ₊₁ (nature) has prob. dist. over its out-edges
     - Nodes in Pᵢ are partitioned into information sets Hᵢ = {hᵢ₁, ..., hᵢⱼ}
   - For each agent i, an action (pure-strategy) sᵢ: Hᵢ → Aᵢ where sᵢ(hᵢⱼ) ∈ Aᵢ(hᵢⱼ)
   - For each agent i, there is a utility function uᵢ(s₁, ..., sₙ)
Normal-Form Representation of Extensive-Form Games

• Any extensive-form game can be transformed into a normal-form game
  • The set of pure strategies of the extensive form = the set of pure strategies in the normal form
  • The set of payoff functions is derived from how combinations of pure strategies result (terminal nodes)

• Every extensive-form game has a unique normal form (opposite is not true)

• The normal-form will suffice to find all the Nash equilibria of the game (most useful when we have two players!)
Other Types or Representations of Games*

<table>
<thead>
<tr>
<th></th>
<th>2, 2</th>
<th>-1, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7, -8</td>
<td>0, 0</td>
<td></td>
</tr>
</tbody>
</table>

*Modified Slides from Vincent Conitzer
Bayesian Games

• We have made an important assumption that the game played is **common knowledge**
  • players are aware of who is playing, what the possible actions of each player are, and how outcomes translate into payoffs (**knowledge of the game is common knowledge**)

• How do we think of situations in which players have **some** idea about their opponents’ characteristics but don’t know for sure what these characteristics are?
Bayesian Games

1. A set of $n$ agents/players

2. For each agent $i$, there is a set of actions $A_i$

3. For each agent $i$, there is a type space $\Theta_i = \{\theta_{i1}, ..., \theta_{ik}\}$

4. For each agent $i$, there is a utility function $u_i : \Theta_i \times A_1 \times ... \times A_n \rightarrow \mathbb{R}$

5. For each agent $i$, there is type belief of other players $\phi_i(\theta_{-i} | \theta_i)$
Bayesian Games

• It is convenient to think about a static **Bayesian game** as one that proceeds through the following steps:
  • **Nature** chooses a profile of types \((\theta_1, \theta_2, \ldots, \theta_n)\)
  • Each player \(i\) learns his own type, \(\theta_i\), which is his **private information**, and then uses his prior \(\phi_i\) to form posterior beliefs over the other types of players.
  • Players **simultaneously** (hence this is a **static game**) choose actions \(a_i \in A_i, \ i \in N\).
  • **Given the players’ choices** \(a = (a_1, a_2, \ldots, a_n)\), the payoffs \(v_i(a; \theta_i)\) are realized for each player \(i \in N\).
Bayesian Games

• For each agent $i$, a pure-strategy $s_i: \Theta_i \rightarrow A_i$

• Pure-strategy Bayesian Nash Equilibrium

A strategy profile $s^* = (s_1^*(\cdot), s_2^*(\cdot), \ldots, s_n^*(\cdot))$ is a pure-strategy Bayesian Nash equilibrium if, for every player $i$, for each of player $i$’s types $\theta_i \in \Theta_i$, and for every $a_i \in A$, $s_i^*(\cdot)$ solves

$$\sum_{\theta_{-i} \in \Theta_{-i}} \phi_i(\theta_{-i}|\theta_i) \nu_i(s_i^*(\theta_i), s_{-i}^*(\theta_{-i}); \theta_i) \geq \sum_{\theta_{-i} \in \Theta_{-i}} \phi_i(\theta_{-i}|\theta_i) \nu_i(a_i, s_{-i}^*(\theta_{-i}); \theta_i).$$

<table>
<thead>
<tr>
<th></th>
<th>CC</th>
<th>CD</th>
<th>DC</th>
<th>DD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C</strong></td>
<td>5, 5</td>
<td>5\lambda, 3\lambda+2</td>
<td>5(1-\lambda), 3\lambda+5</td>
<td>0, 6\lambda+2</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>8, 0</td>
<td>7\lambda+1, 5\lambda-5</td>
<td>8-7\lambda, 1\lambda</td>
<td>1, 6\lambda-5</td>
</tr>
</tbody>
</table>
Other Types or Representations of Games*

**Normal-form games**

<table>
<thead>
<tr>
<th></th>
<th>2, 2</th>
<th>-1, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7, -8</td>
<td>0, 0</td>
<td></td>
</tr>
</tbody>
</table>

**Stochastic games**

<table>
<thead>
<tr>
<th></th>
<th>1, 1</th>
<th>1, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1</td>
<td>0, 0</td>
<td></td>
</tr>
</tbody>
</table>

**Bayesian games**

Row player type 1 (prob. 0.5)

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
</tbody>
</table>

Row player type 2 (prob. 0.5)

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
</tr>
</tbody>
</table>

Column player type 1 (prob. 0.5)

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
</tr>
</tbody>
</table>

Column player type 2 (prob. 0.5)

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
</tr>
</tbody>
</table>

**Stackelberg games (commitment)**

<table>
<thead>
<tr>
<th></th>
<th>2, 1</th>
<th>4, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 0</td>
<td>3, 1</td>
<td></td>
</tr>
</tbody>
</table>

*Modified Slides from Vincent Conitzer*
Stackelberg Games

• Recall the normal-from games

• Now suppose the game is played as follows:
  Player 1 [Leader] commits to playing one of the rows,
  Player 2 [Follower] observes the commitment and then chooses a column

• Stackelberg Equilibrium: What is the optimal strategy for player 1?

<table>
<thead>
<tr>
<th></th>
<th>2, 1</th>
<th>4, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 0</td>
<td>1, 0</td>
<td>3, 1</td>
</tr>
</tbody>
</table>
Roadmap

- Intro
- Classical Game Theory

→ Computational Game Theory

4. Application of Game Theory
Computational Game Theory = CS + Game Theory

• While the area of game theory is originated from the economic literature, computer scientists have made significant contributions to this area from the modeling and computational perspectives in the last decades.

• The new area is known as computational game theory or algorithmic game theory:
  • Representations of Games
  • Computing and Evaluating Equilibrium Concepts
  • Applications of Game Theory
Computational Game Theory = CS + Game Theory

• While the area of game theory is originated from the economic literature, computer scientists have made significant contributions to this area from the modeling and computational perspectives in the last decades.

• The new area is known as computational game theory or algorithmic game theory:
  • Representations of Games
  • Computing and Evaluating Equilibrium Concepts
  • Applications of Game Theory
Recall Normal-form Games

1. A set of n agents/players

2. For each agent i, there is a set of actions (pure-strategies) $S_i$

3. For each agent i, there is a utility function $u_i : S_1 \times \ldots \times S_n \rightarrow \mathbb{R}$

Questions:
What is the representation size? How do we represent $u_i$?

Can we represent some part of the game compactly so that we can perform computation efficiently?
Representing Games*

<table>
<thead>
<tr>
<th></th>
<th>2, 2</th>
<th>-1, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7, -8</td>
<td>0, 0</td>
<td></td>
</tr>
</tbody>
</table>

*normal-form games*

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

*stochastic games*

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

*Bayesian games*

<table>
<thead>
<tr>
<th>U</th>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

*Rational games*

<table>
<thead>
<tr>
<th>U</th>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

*Extensive-form games*

1 gets King
player 1
bet
check
bet
check
player 2
call
fold
call
fold
call
fold

*Resource-graph games*

[Chen & Leyton-Brown, AAMAS’17]
[Chan, Jiang, IJCAI’16, IJCAI’18]
[Chan, Jiang, Keyton-Brown, Mehta, WINE’16]

*Action-graph games*

[Chen & Leyton-Brown, UAI’04]

*Graphical games*

[Kearns, Littman, Singh UAI’01]

*Bayesian games*

[Kearns, Littman, Singh UAI’01]

*Stochastic games*

[Leyton-Brown & Tenenholtz IJCAI’03]

*Normal-form games*

*Modified Slides from Vincent Conitzer*
Graphical Games

• Graphical Games [Kearns et al., 2001] are a compact representation of normal-form games that use graphical models to capture the payoff independence structure of the game.

• Intuitively, a player’s payoff matrix can be written compactly if his payoff is affected only by a subset of the other players.
Example: Road Game

- Consider $n$ agents who have purchased pieces of land alongside a road. Each agent has to decide what to build on his land. His payoff depends on what he builds himself, what is built on the land to either side of his own, and what is built across the road.
Graphical Games

- A graph $G = (N, E)$ - a set of nodes $N$ and edges $E$
  - For each $i \in N$ define the neighborhood relation $\nu: N \to 2^N$ as $\nu(i) = \{i\} \cup \{j \mid (j, i) \in E\}$.

- A graphical game is $(N, E, A, u)$, where:
  - $N$ is a set of $n$ vertices, representing agents;
  - $E$ is a set of undirected edges connecting the nodes $N$;
  - $A = A_1 \times \cdots \times A_n$, where $A_i$ is the set of actions available to agent $i$; and
  - For each $i$, $u_i : A^{\nu(i)} \to \mathbb{R}$, where $A^{\nu(i)} = \prod_{j \in \nu(i)} A_j$.
Graphical Games

• An edge between two vertices $\Rightarrow$ the two agents can affect each other’s payoffs

• Graphical games can represent any game, but not always compactly
  • space complexity is exponential in the size of the largest $v(i)$

• In the road game:
  • the size of the largest $v(i)$ is 4, independent of the total number of agents
  • the representation requires space polynomial in $n$, while a normal-form representation requires space exponential in $n$
Representing Games*

<table>
<thead>
<tr>
<th></th>
<th>2, 2</th>
<th>-1, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7, -8</td>
<td>0, 0</td>
<td></td>
</tr>
</tbody>
</table>

normal-form games

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

row player type 1 (prob. 0.5)

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

row player type 2 (prob. 0.5)

column player type 1 (prob. 0.5)

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Stochastic games

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Bayesian games

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

action-graph games

[Chen, Jiang, Leyton-Brown AAAI'17]
[Chan, Jiang, IJCAI'16, IJCAI'18]
[Chan, Jiang, Keyton-Brown, Mehta, WINE'16]

*Modified Slides from Vincent Conitzer
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 might be better ex ante
  • utility also depends on the number of other players who choose the same or an adjacent location
Action-graph Games

- An action-graph game (AGG) is a tuple \((N, A, (A, E), u)\) where
  - \(N\) is the set of agents;
  - \(A = A_1 \times \cdots \times A_n\), where \(A_i\) is the set of actions available to agent \(i\);
  - \((A, E)\) is an action graph, where \(A = \bigcup_{i \in N} A_i\) is the set of distinct actions; and
  - \(u = \{u_\alpha | \alpha \in A\}\), \(u_\alpha : C(\alpha) \rightarrow \mathbb{R}\).
Action-graph Games

- AGGs can represent any game.
- Overall, AGGs are more compact than the normal form.
  - In the Road game, since each node has at most four incoming edges, we only need to store $O(n^4)$ numbers for each node, and $O(|A|n^4)$ numbers for the entire game.
  - In general, when the in-degree of the action graph is bounded by a constant, the space complexity of the AGG representation is polynomial in $n$.
- When max in-degree $I$ is bounded by a constant:
  - Polynomial size: $O(|A|n^I)$
  - Size of normal form is: $O(n|A|^n)$
Representing Games*

- **normal-form games**
  - \[
  \begin{array}{cc}
  2, 2 & -1, 0 \\
  -7, -8 & 0, 0 \\
  \end{array}
  \]

- **stochastic games**
  - \[
  \begin{array}{c}
  2, 2 \\
  0, 3 \\
  3, 0 \\
  1, 1 \\
  \end{array}
  \]

- **extensive-form games**
  - 1 gets King, 1 gets Jack
  - player 1
  - bet, check, bet
  - player 2
  - call, fold, call, call, fold, call, fold

- **Bayesian games**
  - row player
    - type 1 (prob. 0.5)
      - U: 4, 6
      - D: 2, 4
  - column player
    - type 1 (prob. 0.5)
      - U: 4, 6
      - D: 2, 2

- **Bayesian games**
  - row player
    - type 2 (prob. 0.5)
      - U: 2, 4
      - D: 4, 2
  - column player
    - type 2 (prob. 0.5)
      - U: 2, 2
      - D: 4, 2

- **graphical games**
  - [Kearns, Littman, Singh UAI’01]

- **action-graph games**
  - [Leyton-Brown & Tennenholtz IJCAI’03]
  - [Bhat & Leyton-Brown, UAI’04]
  - [Jiang, Leyton-Brown, Bhat GEB’11]

- **resource-graph games**
  - [Jiang, Chan, Leyton-Brown, AAAI’17]
  - [Chan, Jiang, IJCAI’16, IJCAI’18]
  - [Chan, Jiang, Keyton-Brown, Mehta, WINE’16]

*Modified Slides from Vincent Conitzer
Many Games with an Exponential Number of Actions!!

Attacker: Select 1 target to attack

Defender: Select k targets to defend

Question: Is there a compact representation that can capture all of these games?
Resource Graph Games

- A set, \( N = \{1, 2, \ldots, n\} \), of players
- A set, \( A = \{1, 2, \ldots, m\} \), of resources

- Player \( i \)'s strategy set, \( S_i \), consists of subsets of resources \( i \) can use (represented by binary vectors)

- \( G = (A, E) \) is a directed resource graph

- Given \( s = (s_i, s_{-i}) \in S_1 \times \ldots \times S_n \), i’s utility is
  \[
  u_i(s_i, s_{-i}) = \sum_{r \in A} s_{ir} u^r(#(\text{neighbors}(r), s))
  \]

  \( #(\text{neighbors}(r), s) \) counts number of times the neighbors of \( r \) is used under \( s \)

E.g. \( \text{neighbors}(2) = \{1, 2\} \)
Polytopal Strategy Spaces

• The strategy space of each player can be represented compactly using a polytope.

• For each player $i$, his/her strategy set

$$S_i = P_i \cap \{0, 1\}^m$$

$$P_i = \{ x \in \mathbb{R}_+^m \mid D_i x \leq f_i \}$$

where $D_i \in \mathbb{Z}_+^{l_i \times m}$ and $f_i \in \mathbb{Z}_+^{l_i}$

$P_i$ is a rational polytope defined by $l_i$ linear constraints!!!
Resource Graph Games (RGGs)

- A set, \( N = \{1, 2, \ldots, n\} \), of players
- A set, \( A = \{1, 2, \ldots, m\} \), of resources

Player \( i \)'s strategy set \( S_i = \{ x \mid D_i x \leq f_i \} \cap \{0, 1\}^m \)

\[ G = (A, E) \] is a directed resource graph

Given \( s = (s_i, s_{-i}) \in S_1 \times \ldots \times S_n \), i's utility is

\[ u_i(s_i, s_{-i}) = \sum_{r \in A} s_{ir} u^r(\#(\text{neighbors}(r), s)) \]

\( \#(\text{neighbors}(r), s) \) counts number of times the neighbors of \( r \) is used under \( s \)
Computational Game Theory = CS + Game Theory

• While the area of game theory is originated from the economic literature, computer scientists have made significant contributions to this area from the modeling and computational perspectives in the last decades

• The new area is known as computational game theory or algorithmic game theory
  • Representations of Games
  • Computing and Evaluating Equilibrium Concepts
  • Applications of Game Theory
Computational Questions

• Given a game (along with its representations) and an equilibrium concept (i.e., Nash equilibrium):

  • (Existence) Does the equilibrium exist? If so, find one or all.

  • (Guaranteed Properties) Does there exist an equilibrium in which, e.g.,
    • (Social welfare) the social welfare is at least k?
    • (Payoff) player i obtains an expected payoff of at least v?
    • (Uniqueness) a unique equilibrium in G?
Computational Questions for Normal-form Games

• Given a game (= normal-form) and an equilibrium concept (= pure-strategy Nash equilibrium):
  
  • (Existence) Does the equilibrium exist? If so, find one or all.
  • (Guaranteed Properties) Does there exist an equilibrium in which, e.g.,
    • (Social welfare) the social welfare is at least k?
    • (Payoff) player i obtains a payoff of at least v?
    • (Uniqueness) a unique equilibrium in G?

They all can be determined/computed in polynomial time w.r.t. to the representation size!!!
Computational Questions for Graphical Games

- Given a game (= graphical) and an equilibrium concept (= pure-strategy Nash equilibrium):
  - *(Existence)* Does the equilibrium exist? If so, find one or all.
  - *(Guaranteed Properties)* Does there exist an equilibrium in which, e.g.,
    - *(Social welfare)* the social welfare is at least $k$?
    - *(Payoff)* player $i$ obtains a payoff of at least $v$?
    - *(Uniqueness)* a unique equilibrium in $G$?

They are all NP-complete!
Computational Questions for Action-graph Games

• Given a game (= action-graph) and an equilibrium concept (= pure-strategy Nash equilibrium):
  
  • (Existence) Does the equilibrium exist? If so, find one or all.
  • (Guaranteed Properties) Does there exist an equilibrium in which, e.g.,
    • (Social welfare) the social welfare is at least k?
    • (Payoff) player i obtains a payoff of at least v?
    • (Uniqueness) a unique equilibrium in G?

They are all NP-complete!
Computational Questions for Normal-form Games

- Given a game (= normal-form) and an equilibrium concept (= mixed-strategy Nash equilibrium):

  - (Existence) Does the equilibrium exist? If so, find one or all.
    - Yes! [Nash, 1950]
2-Player General-sum Games

• Complexity was open for a long time
  – [Papadimitriou STOC01]: “together with factoring [...] the most important concrete open question on the boundary of P today”

• Recent sequence of papers shows that computing one (any) Nash equilibrium is PPAD-complete (even in 2-player games) [Daskalakis, Goldberg, Papadimitriou 05; Chen, Deng 05]

• All known algorithms require exponential time (in the worst case)
Computational Questions for Normal-form Games

• Given a game (= Normal-form) and an equilibrium concept (= mixed-strategy Nash equilibrium):

  • (Existence) Does the equilibrium exist? If so, find one or all.
    • Yes! [Nash, 1950]
    • The problem of finding a NE of a general game with two or more players is PPAD-complete
    • Since this holds for two players, it holds for any representations discussed so far

  • (Guaranteed Properties) Does there exist an equilibrium in which, e.g.,
    • (Social welfare) the social welfare is at least k? NP-complete
    • (Payoff) player i obtains an expected payoff of at least v? NP-complete
    • (Uniqueness) a unique equilibrium in G? NP-complete
Computational Questions

• So, what is next since many of the computational questions are hard?

  • Study games with special structures

  • Look at brute force algorithms to find equilibria

  • Look at efficient algorithms and heuristics to compute approximate equilibria

  • Study alternative solution concepts
Computational Questions

• So, what is next since many of the computational questions are hard?
  • Study games with special structures
    • Zero sum games, graphical games, and action-graph games
  • Look at brute force algorithms to find equilibria
  • Look at efficient algorithms and heuristics to compute approximate equilibria
  • Study alternative solution concepts
2-player Zero Sum Games

1. 2 agents/players

2. For each agent $i$, there is a set of actions (pure-strategies) $A_i$

3. For each agent $i$, there is a utility function $u_i : A_1 \times A_2 \rightarrow \mathbb{R}$

4. For any $(a_1, a_2)$, $u_1(a_1, a_2) + u_2(a_1, a_2) = 0$

A mixed-strategy Nash equilibrium can be computed in polynomial time using linear programming (Minimax Theorem)!
Maximin Strategy and Minimax Strategy (for player 1)

**Definition 3.2** (Maximin Strategy). A maximin strategy $\bar{s}_1$ for player 1 in a two-player game is a strategy that satisfies:

$$\bar{s}_1 \in \arg \max_{s_1} \left[ \min_{k \in A_2} u_1(s_1, k) \right].$$

(3.1)

A maximin strategy, perhaps randomized, for player 1 maximizes player 1’s expected utility $u_1(s_1, k)$, given that player 2, with knowledge of player 1’s strategy, selects an action $k$ to minimize player 1’s expected utility.

**Definition 3.3** (Minimax Strategy). A minimax strategy $s_1$ for player 1 in a two-player game is any strategy that satisfies:

$$s_1 \in \arg \min_{s_1} \left[ \max_{k \in A_2} u_2(s_1, k) \right].$$

(3.2)
Maximin Strategy and Minimax Value

Let $\bar{v}_1$ (the maximin value) denote

$$\bar{v}_1 = \min_{k \in A_2} u_1(\bar{s}_1, k),$$  \hspace{1cm} (3.3)

and the expected payoff to player 1 when it plays a maximin strategy. Let $v_1$ (the minimax value) denote

$$v_1 = \max_{j \in A_1} u_1(j, s_2),$$  \hspace{1cm} (3.4)

and the expected payoff to player 1 when it plays a best response to a minimax strategy by player 2.
2-Player Zero-sum Games

Theorem 3.1 (Minimax Theorem). In any two-player, zero-sum game,

(1) For each player, the set of maximin strategies is identical to the set of minimax strategies.

(2) Any maximin or minimax strategy for player 1 and any maximin or minimax strategy for player 2 is a Nash equilibrium, and these correspond to all Nash equilibria.

(3) Each player’s maximin value is equal to its minimax value, and equal to its expected utility in any Nash equilibrium.

Because of this correspondence, we can compute a Nash equilibrium in a two-player, zero-sum game by finding a maximin strategy for each player.
2-Player Zero-sum Games: Linear Programming (Maximin for P1)

\[ LP_1 : \max_{v_1, x} v_1 \]

s.t. \[ \sum_{j \in A_1} u_1(j, k) \cdot x_j \geq v_1, \quad \forall k \in A_2 \]  \hspace{1cm} (3.6)

\[ \sum_{j \in A_1} x_j = 1 \]  \hspace{1cm} (3.7)

\[ x_j \geq 0, \quad \forall j \in A_1 \]  \hspace{1cm} (3.8)

• \( u_1(.) \) are constants, \( x_j \) and \( v_1 \) are variables
• (3.6) \( \rightarrow \) P1’s expected utility given P2’s actions
• (3.7) and (3.8) \( \rightarrow \) valid distributions or mixed strategies
Graphical Games

• Deciding whether a graphical game has a pure-strategy Nash Equilibrium is in P for all classes of games with bounded treewidth or hypertreewidth. [Gottlob et al. 2003, 2005] [Daskalakis and Papadimitriou, 2006]

• When the graph (N, E) is a path or a cycle, a message-passing algorithm can find an exact equilibrium in polynomial time [Elkind, Goldberg & Goldberg, 2006]

• When the graph (N, E) is a tree, a message-passing algorithm can compute an ε-Nash equilibrium in time polynomial in $1/\varepsilon$ and the size of the representation [Kearns et al., 2001]
Action-Graph Games

• For symmetric AGGs with bounded treewidth, existence of a pure-strategy Nash Equilibrium can be determined in polynomial time. [Jiang & Leyton-Brown, 2007]

• There exists a fully polynomial time approximation scheme for computing a mixed-strategy Nash Equilibrium of AGGs with constant degree, constant treewidth and a constant number of distinct action sets (but unbounded number of actions)
Computational Questions

• So, what is next since many of the computational questions are hard?
  • Study games with special structures
    • Zero sum games, graphical games, and action-graph games
  • Look at brute force algorithms to find equilibria
  • Look at efficient algorithms and heuristics to compute approximate equilibria
  • Study alternative solution concepts
Brute Force Algorithms

• 2-player games
  • Support enumerations
  • Linear complementarity programming
  • Lemke–Howson algorithm

• N-player games
  • Simplicial Subdivision
  • Govindan-Wilson
  • Constraint satisfaction problem (CSP) based algorithms

Gambit
Software Tools for Game Theory
Computational Questions

• So, what is next since many of the computational questions are hard?
  
  • Study games with special structures
    • Zero sum games, graphical games, and action-graph games
  
  • Look at brute force algorithms to find equilibria
  
  • Look at efficient algorithms and heuristics to compute approximate equilibria
    • Learning in games
  
• Study alternative solution concepts
Learning in Games

• Approach we have taken so far when playing a game: just compute an optimal/equilibrium strategy
• Another approach: learn how to play a game by
  • playing it many times, and
  • updating your strategy based on experience
• Why?
  • Some of the game’s utilities (especially the other players’) may be unknown to you
  • The other players may not be playing an equilibrium strategy
  • Computing an optimal strategy can be hard
  • Learning is what humans typically do
  • ....
• Learning strategies - strategies for the repeated game
  • Fictitious play, no-regret Learning algorithms, targeted learning
• Does learning converge to equilibrium?
Fictitious Play [Brown 1951]

• Fictitious play is an instance of model-based learning, in which the learner explicitly maintains beliefs about the opponent's strategy.

• The structure of such techniques is straightforward.

Initialize beliefs about the opponent’s strategy

repeat

  Play a best response to the assessed strategy of the opponent

  Observe the opponent’s actual play and update beliefs accordingly
Fictitious Play [Brown 1951]

Initialize beliefs about the opponent’s strategy
repeat
  Play a best response to the assessed strategy of the opponent
  Observe the opponent’s actual play and update beliefs accordingly

• For example, if $A$ is the set of the opponent’s actions, and for every $a \in A$, we let $w(a)$ be the number of times that the opponent has played action $a$, then the agent assesses the probability (or empirical distribution) of $a$ in the opponent’s mixed strategy as

$$P(a) = \frac{w(a)}{\sum_{a' \in A} w(a')}$$
Fictitious Play [Brown 1951]

• In the first round, play something arbitrary
• In each following round, play a best response against the empirical distribution of the other players’ play
  • I.e., as if other player randomly selects from his past actions
• Again, if this converges, we have a Nash equilibrium
• Can still fail to converge...
Fictitious play is guaranteed to converge in...

- Two-player zero-sum games [Robinson 1951]
- Generic 2x2 games [Miyasawa 1961]
- Games solvable by iterated strict dominance [Nachbar 1990]
- Weighted potential games [Monderer & Shapley 1996]
- Fictitious play always converges to the set of $\frac{1}{2}$-approximate equilibria [C. 2009; more detailed analysis by Goldberg, Savani, Sørensen, Ventre 2011]
- Not in general (Shapley 1964)
Computational Questions

• So, what is next since many of the computational questions are hard?

  • Study games with special structures
    • Zero sum games, graphical games, and action-graph games

  • Look at brute force algorithms to find equilibria

  • Look at efficient algorithms and heuristics to compute approximate equilibria
    • Learning in games

  • Study alternative solution concepts
    • Correlated equilibrium and coarse correlated equilibrium (they are usually easy to compute using LP based techniques)
Price of Anarchy (PoA)

- We know that equilibrium behavior can lead to suboptimal outcomes (prisoners’ dilemma)
- Price of anarchy (PoA) quantifies the relative quality of the outcome in equilibrium to the quality of the best possible outcome in the worst case over all possible equilibria.

Let $NE(G)$ denote the set of pure-strategy Nash equilibria of a game $G$ and $C$ be a maximizing objective.

$$PoA(\Gamma) = \max_{a^* \in NE(\Gamma)} \left[ \frac{W(a^{opt})}{W(a^*)} \right]$$

- The goal is to derive lower and upper bounds to PoA.
Price of Anarchy (PoA)

- Direct Analysis
- Smoothness Framework
- Linear Programming Duality Theory
Computational Game Theory =
CS + Game Theory

• While the area of game theory is originated from the economic literature, computer scientists have made significant contributions to this area from the modeling and computational perspectives in the last decades

• The new area is known as computational game theory or algorithmic game theory
  • Representations of Games
  • Computing and Evaluating Equilibrium Concepts
  • Applications of Game Theory
Roadmap

- Intro
- Classical Game Theory
- Computational Game Theory
- Application of Game Theory
Game theory has numerous applications!

Q: What is game theory? How can we use it to model these real-world scenarios?
Game Theory Modeling and Application Research

• **Step 1**: Think about your problem domain (e.g., identify players, actions, and utility functions); the utility functions should be parameterized with parameters and depend on agents’ actions.

• **Step 2**: Identify an equilibrium concept and look at the computational and/or evaluation questions (hardness results and/or algorithms).

• **Step 3**: Conduct simulations to highlight equilibrium behaviors using real-world parameters or instances.
Variants of Security Games Used and Deployed in Real World

- Infrastructure Security Games
  - LAX
  - Coast Guard
  - TSA
  - LA Sheriff

- Green Security Games

- Opportunistic Crime Games

- Cyber Security Games

[Teamcore]
Stackelberg Security Games

Attackers use surveillance in planning attacks

Defender commits to a mixed strategy
ARMOR: Deployed at LAX 2007

- “Assistant for Randomized Monitoring Over Routes”
  - Problem 1: Schedule vehicle checkpoints
  - Problem 2: Schedule canine patrols
- Randomized schedule: (i) target weights; (ii) surveillance
ARMOR Canine: Interface
Federal Air Marshals Service (FAMS)

Undercover, in-flight law enforcement

Flights (each day)
~27,000 domestic flights
~2,000 international flights

Not enough air marshals:
Allocate air marshals to flights?
Federal Air Marshals Service (FAMS)

- Massive scheduling problem
- Adversary may exploit predictable schedules
- Complex constraints: tours, duty hours, off-hours

100 flights, 10 officers: 

$1.7 \times 10^{13}$ combinations

Overall problem: 30000 flights, 3000 officers

*Our focus: international sector*
IRIS: Intelligent Randomization in International Scheduling (Deployed 2009)
PROTECT (Boston, NY and Beyond)

- US Coast Guard: Port Resilience Operational / Tactical Enforcement to Combat Terrorism
- Randomized patrols; deployed in Boston & NY, with more to follow
- More realistic models of human behaviors
Motivation: Many targets, few resources

How to assign limited resources to defend the targets?

*Game Theory*
Stackelberg Security Games

- Only attack a single Terminal

We need to compactly represent defender strategies so that we can perform efficient equilibrium computation!!!!

Terminal 1

Terminal 2

Terminal T

- Has limited resources

Defender has exponentially many strategies!
Attackers use surveillance in planning attacks

Defender commits to a mixed strategy

Main Contributions: (1) Extending Stackelberg games to handle exponential numbers of strategies, (2) introducing efficient algorithms to compute Stackelberg equilibria, (3) experimental evaluations of the algorithms, and (4) real-world deployments
Domain Specific Game-theoretic Models

- Interdependent Security Games
- Influence Games
- Influence Maximization Games
- Public Good Contribution Games
- Schelling Games
- Voting Games
- Many more ... and in other CS areas

Main Contributions: (1) Introducing game-theoretic models for their domains, (2) analyzing and computing (Nash) equilibria, and (3 optional) experimental evaluations of the algorithms
Recommended Materials

• Book:
  • Game Theory: An Introduction
  • Essentials of Game Theory
  • Multiagent Systems Algorithmic, Game-Theoretic, and Logical Foundations
  • Economics and Computation

• Tutorials:
  • Computational Game Theory and Its Applications