Game Theory with Costly Computation

Rafael Pass

Joe Halpern

Cornell
A Computational Game

You are given random odd n-bit number (n big)

You either **GIVE UP**, or guess **PRIME/NOT PRIME**

Payoff:
- **GIVE UP**: $10
- **Correct answer**: $1000
- **Wrong answer**: -$10000n

What do you do? Depends on cost of comp!
Add Computation into Game Theory

Idea goes back to Herbert Simon ’55 “bounded rationality”

Two lines of study:

• restricted strategies of player [Neyman’85, MW’86, ..,PY’94, UV’99, DHR’00]
• charging for size of strategy [Rubinstein’87, …,BKK’06]

Our goal: provide a general model, investigate its properties
Games

Players 1,…,m

Available Actions

Utility

Strategy σ

Expected Utility

Nash Eq  \( (\sigma_1, ..., \sigma_m) \) s.t  \( \forall i \forall \sigma'_i U_i(\sigma_i, \sigma_{-i}) \geq U_i(\sigma'_i, \sigma_{-i}) \)

“If others are playing their strategies, I better stick to mine!”

Always exists! [N51]

u_i(a_1,…,a_m)

1 \( \ldots \) m

Rock  Paper  Scissors

\( U_i(\sigma_i, \sigma_{-i}) = \text{Exp}[u_i(a)] \)
Bayesian Games

Types (private information)

Strategy

Actions

Utility
t = odd n-bit number (n big)

Actions: GIVE UP, PRIME, NOT PRIME

Payoff:
- GIVE UP: $10
- Correct answer: $1000
- Wrong answer: -$10000n

Only NE
Bayesian Machine Games

Strategy = randomized TM

Complexity function \texttt{complex}: M \times \{0,1\}^* \rightarrow \mathbb{N}

- \texttt{complex}(M,v) = \text{complexity of } M \text{ on view } v = \text{t ; r}
- (e.g., time, space, size, communication...)

Utility depends on:
- types
- actions
- complexities of machines
Bayesian Machine Game ([m], Δ, C, u):

Types

Strategy

Actions

Utility

\[ \Delta \]

\[ M_1(t_1; r_1) \]

\[ a_1 \]

\[ u_1(t, \hat{a}, \hat{c}) \]

\[ t_1 \]

\[ t_{m} \]

\[ M_m(t_m; r_m) \]

\[ a_m \]

\[ u_m(t, \hat{a}, \hat{c}) \]

\[ c_1 = \text{complex}(M_1, (t_1; r_1)) \]

\[ c_m = \text{complex}(M_m, (t_m; r_m)) \]
Nash Eq in Machine Games

As before:

\[(M_1, ..., M_m) \text{ s.t } \forall i \forall M'_i U_i(M_i, M_{-i}) \geq U_i(M'_i, M_{-i})\]

But do they ALWAYS exist?

NO!
Roshambo with Costly Comp

Utility as usual but **subtract** # of coin tosses

<table>
<thead>
<tr>
<th></th>
<th>Rock</th>
<th>Paper</th>
<th>Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td>(0,0)</td>
<td>(1,2)</td>
<td>(2,1)</td>
</tr>
<tr>
<td>Paper</td>
<td>(2,1)</td>
<td>(0,0)</td>
<td>(1,2)</td>
</tr>
<tr>
<td>Scissors</td>
<td>(1,2)</td>
<td>(2,1)</td>
<td>(0,0)</td>
</tr>
</tbody>
</table>

Assume exist NE such that is randomizing

⇒ exist det. strategy that does better

same original payoff, better complexity

Only possible NE is det.

But there are no deterministic NE!
Strange? Natural?
The World Champion
Why? Coin tossing is hard?
Nash Eq in Machine Games

**Thm:** Every finite *computable* machine game (i.e., utilities and probabilities are computable) where “randomization is free” has a NE.

**Main Lemma [Computational Analog of Nash Thm]:** Bayesian games where $u_i$ and $\Delta$ are *computable*, have a NE that is *implementable by randomized Turing Machines* terminating with probability 1.
Explaining Observed Behavior

There exist many “paradoxical” games where traditional GT solutions concepts provide the “wrong” answer:

- repeated prisoner’s dilemma [LR’51]
- first-impression’s matter bias [R’99]
- belief polarization [LRL’79]
- use of “bad” randomness in sports competition [WW’01]
- ...

One remedy: behavioral economics [KT’81]

Use psychology or models of brain to explain “irrational” behavior (at a qualitative level)
Secure Computation \([Y,GMW]\)

\(m\) parties, each with private input \(x_i\)

Goal: \textit{secure compute} function \(F\)

\\[
\text{Election:} \\
F(x_1, \ldots, x_m) = \text{tally}
\]

(M) \textit{securely computes} \(F\) if it provides the same \textit{privacy} and \textit{correctness} guarantees as if a trusted party had computed \(F\)

\(-\) Formalized using \textit{ZK simulation} \([GMR]\)

Does this mean players \textit{want} to run (M)?
Our Goal: capture intuition that $(M)$ securely implements $F$ if

**WHenever**: players **WANT** to compute $F$ using their inputs

**Then**: players **WANT** to run $(M)$ on their inputs
Our Goal: capture intuition that

(M) securely implements F if

In every situation where:
players WANT to compute F using their inputs

It holds that:
players WANT to run (M) on their inputs

<table>
<thead>
<tr>
<th>situation</th>
<th>= game</th>
</tr>
</thead>
<tbody>
<tr>
<td>WANT</td>
<td>“is a NE”</td>
</tr>
</tbody>
</table>
(M) universally implements F if:

In every machine game G where:
providing true input to a trusted party computing F and outputting what the trusted party replies, is a NE in G.

It holds that:
running (M) on true inputs is a NE in G.

Notes:
• similar to [Forges’87, ILM’06] but with costly comp
• guarantees that (M) does not “leak more” information than F
The Theorem

Tight connection between cryptographic notion of secure computation and universal implementation.

The notions are “essentially” equivalent.

Nash equilibrium and “ZK simulation”
are intimately connected
Framework for GT with Costly Computation

- Give simple computational explanations observed behavior in “paradoxical” games.
  - Can we use behavioral experiments to determine “cost of computation”?

- **Nash Equilibrium** v.s. **Sequential Equilibrium**

- We have assumed that players “understand the game” (i.e., they know how well a machine does, and what its complexity is).
  - Can also model players that have “beliefs” about how well a machine does.
  - But computing these beliefs might itself be costly!

**GT definition of security**

- “Equivalent” to secure computation in the most general setting,
- But helpful in circumventing lower-bound by considering restricted classes of games (e.g., players strictly prefer to compute less, or don’t want to be caught cheating)
– Alice and Bob want to find out if they love each other: compute **AND** of their inputs.

– **Desiderata:**
  - Want to know the output.
  - Don’t want reveal input.
  - If I get the output, (slightly) prefer to trick you.

– [Shoham-Tennenholtz’03] **Impossible** even with a trusted party computing **AND**!
  - If I have input 0, always better to say 1 to trusted party (since I already know the output).
- Use a crypto protocol, where players need to solve a “computational riddle” [DN] if they use input 1.
  - But still “indistinguishable” if you use input 0 or 1.
- Rational to provide true input if:
  - Cost of solving riddle > gain to trying to trick other player.
  - Value of Privacy > cost of solving riddle