Microsoft Research

Each year Microsoft Research hosts hundreds of influential speakers from around the world including leading scientists, renowned experts in technology, book authors, and leading academics, and makes videos of these lectures freely available.

2016 © Microsoft Corporation. All rights reserved.
AI for Large Imperfect-Information Games: Beating Top Humans in No-Limit Poker

Noam Brown
Computer Science Department
Carnegie Mellon University

Joint work with my advisor Tuomas Sandholm
Imperfect-Information Games
Imperfect-Information Games
Imperfect-Information Games

[Images with red crosses over them]
Imperfect-Information Games
Heads-Up No-Limit Texas Hold’em

- Has become the main **benchmark and challenge problem** in AI for imperfect-information games
- No-Limit Betting = Continuous Action Space
  - Technically, $10^{161}$ situations since bets must be integers
- The most popular variant of poker in the world
  - Played in the World Series of Poker Main Event
  - Featured in *Casino Royale* and *Rounders*
- “Purest form of poker”
- No prior AI has been able to beat top humans
2017 Brains vs AI

- Libratus (our 2017 AI) against four of the best heads-up no-limit Texas Hold’em specialist pros

- 120,000 hands over 20 days in January 2017
- $200,000 divided among the pros based on performance
- Conservative experiment design
How good are these pros?

I don't follow pro poker like I once did, my hero's still being Ivey, Hellmuth(hero?), Ferguson, Dwan, Doyle... the guys you could find on every poker show from the mid 2000's. Who can enlighten me on how good these pros are in comparison, to say, the megastars of poker? If Ivey and Hellmuth were to go head to head against this AI, do you think they would have a better shot? Or, are the players playing against the AI a little less than stellar? Poker Pro seems kind of subjective these days.

[–] dabsindenver 11 points 1 year ago

These players would absolutely trounce all of the 2000s heroes in heads up poker. The hero's from the 2000s would be division III college players while these guys are all-star caliber pros. These guys are top pros in the heads up world and make their living playing it and similar games.
Final Result

- Libratus beat the top humans in this game by a lot
  - 147 mbb/game
  - Statistical significance 99.98%, i.e., p-value of 0.0002
  - Each human lost to Libratus
Lengpudashi vs humans event

- 36,000 hands against 6 Chinese poker players
  - WSOP bracelet winner
  - Expertise in computer science & ML
  - Studied Libratus’s hand histories in advance

- Lengudashi won by 220 mbb/game
  - Won each of the 9 sessions
  - Also beat each human individually
  - Watched live by millions of people
Why are imperfect-information games hard?
Why are imperfect-information games hard?

Because an optimal strategy for a subgame cannot be determined from that subgame alone.
Real-time planning is important

![Bar chart showing Elo ratings for different models, with 'Full AlphaGo Zero' and 'No real-time planning' highlighted.](image)
Perfect-Information Games
Perfect-Information Games
Perfect-Information Games
Perfect-Information Games
Perfect-Information Games
Imperfect-Information Games
Imperfect-Information Games
Imperfect-Information Games
Imperfect-Information Games
Nash Equilibrium

- **Nash Equilibrium**: a profile of strategies in which no player can improve by deviating

- In two-player zero-sum games, playing a Nash equilibrium ensures you will not lose in expectation

- **Exploitability**: Worst-case performance relative to Nash
Why care about exploitability?

Real-world AI must be **robust** to adversarial adaptation and exploitation

Fan Hui vs AlphaGo

OpenAI 1v1 Dota2 Matches
Example game: Coin Toss

\[
P = 0.5 \quad \text{Heads} \quad P = 0.5 \quad \text{Tails}
\]
Example game: Coin Toss
Example game: Coin Toss
Example game: Coin Toss
Example game: Coin Toss
Example game: Coin Toss

- P1 Information Set
- P2 Information Set
- C
- P = 0.5
- EV = 0.5
- EV = -0.5
- P1
- P2
- Heads
- Tails
- Play
- Self
- -1
- 1
- 1
- -1
Example game: Coin Toss

- EV = 0.5
- P = 0.5
- Heads
- Play

- EV = -0.5
- P = 0.5
- Tails
- Play

Imperfect-Information Subgame

P1

P2

P2

Heads

Tails

-1

1

1

-1
Example game: Coin Toss
Example game: Coin Toss

Heads EV = 0.5
Tails EV = 1.0
Average = 0.75
Example game: Coin Toss

EV = 0.5
Sell
Play

EV = -0.5
Sell
Play

P = 0.5
Heads
P = 0.5
Tails

P = 0.0
Heads
P = 1.0
Tails

P = 0.0
Heads
P = 1.0
Tails

-1
1
1
1
-1
Example game: Coin Toss
Example game: Coin Toss

Heads EV = 1.0
Tails EV = -0.5
Average = 0.25
Example game: Coin Toss

EV = 0.5

P1
Play

P2

-1

1

1

-1

P = 0.25

Heads

Tails

P = 0.75

P = 0.5

Heads

Tails

P = 0.75

P = 0.5

Sell

Play
Example game: Coin Toss

Heads EV = 0.5
Tails EV = -0.5
Average = 0.0
Example game: Coin Toss
Example game: Coin Toss

Determining the optimal strategy in the “Play” subgame requires knowledge of the values the opponent could receive in other subgames.
Example game: Coin Toss

[Idea: Estimate the values the opponent receives for actions in an equilibrium]

[Burch et al. AAAI-14, Moravcik et al. AAAI-16, Brown & Sandholm NIPS-17]
Example game: Coin Toss

[Brown & Sandholm Science-17]

Theorem: If estimates of opponent values are off by at most $\delta$, then safe subgame solving has at most $\delta$ exploitability.
Nested subgame solving
Nested subgame solving
Nested subgame solving
Nested subgame solving
Nested subgame solving
Nested subgame solving
Unsafe Subgame Solving

- Estimate the opponent’s strategy
- This gives a belief distribution over states
- Update beliefs via Bayes Rule

[Canzfried & Sandholm AAMAS 2015]
Nested subgame solving
Unsafe Subgame Solving

- Estimate the opponent’s strategy
- This gives a belief distribution over states
- Update beliefs via Bayes Rule

Diagram: A game tree with nodes and edges labeled with probabilities and outcomes. The tree shows nodes P1, P2, C, and connections with branches labeled 'Heads' and 'Tails' with associated probabilities and expected values (EV).

References: Ganzfried & Sandholm (AAMAS 2015)
Unsafe Subgame Solving

[Ganzfried & Sandholm AAMAS 2015]

- Estimate the opponent’s strategy
- This gives a belief distribution over states
- Update beliefs via Bayes Rule
Unsafe Subgame Solving

- Estimate the opponent’s strategy
- This gives a belief distribution over states
- Update beliefs via Bayes Rule
Unsafe Subgame Solving

[Ganzfried & Sandholm AAMAS 2015]

- Estimate the opponent’s strategy
- This gives a belief distribution over states
- Update beliefs via Bayes Rule

---

Diagram:

- Node C
  - P1: Heads (P = 0.5), Tails (P = 0.5)
  - P1: Heads (P = 0.2), Sell (P = 0.8)
    - EV = 0.5
  - P1: Tails (P = 0.7)
    - EV = -0.5

- Node P2
  - Heads (P = 0.3), Play (P = 0.7)
  - EV = -0.5

- Node P2
  - Heads (P = 0.3), Tails (P = 0.7)
  - EV = 1

- Edge probabilities:
  - 73% Probability
  - 27% Probability

- States:
  - 1

---
Unsafe Subgame Solving

[Ganzfried & Sandholm AAMAS 2015]

- Estimate the opponent’s strategy
- This gives a belief distribution over states
- Update beliefs via Bayes Rule

![Game Tree Diagram]

- P1
  - Heads: P = 0.2, Sell: EV = 0.5
  - Tails: P = 0.8, Play: EV = -1.0

- P2
  - Heads: P = 1.0, Purchase: EV = 1.0
  - Tails: P = 0.0, Sell: EV = -1.0

73% Probability
27% Probability
Unsafe Subgame Solving

We must account for the opponent's ability to adapt!
But, in practice, unsafe solving works pretty well sometimes
Safe Subgame Solving

[Burch et al. AAAI-14, Moravcik et al. AAAI-16, Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

(Brown & Sandholm NIPS-17)
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

[Brown & Sandholm NIPS-17]
Reach subgame solving

For off-path actions, we must consider how applying subgame solving would have changed their EVs if those actions were chosen instead.

Solution: split “slack” among actions by actions’ probabilities.

**Theorem:** Reach subgame solving will not increase a safe technique’s exploitability, and lower it if there is slack.
Experiments on medium-sized games

• Our best reach subgame solving technique has $3x$ less exploitability than the best prior safe subgame-solving technique.

• Nested solving is $12x$ less exploitable than techniques that do not use real-time reasoning.
Reach subgame solving

[Brown & Sandholm NIPS-17]

- For off-path actions, we must consider how applying subgame solving would have changed their EVs if those actions were chosen instead.
- Solution: split “slack” among actions by actions’ probabilities.

**Theorem:** Reach subgame solving will not increase a safe technique’s exploitability, and lower it if there is slack.
Experiments on medium-sized games

- Our best **reach subgame solving** technique has **3x** less exploitability than the best prior safe subgame-solving technique.

- **Nested solving** is **12x** less exploitable than techniques that do not use real-time reasoning.
Why are imperfect-information games hard?

1) Because an optimal strategy for a subgame cannot be determined from that subgame alone.
Why are imperfect-information games hard?

1) Because an optimal strategy for a subgame cannot be determined from that subgame alone

2) Because states don’t have well-defined values
Perfect-Information Games and Single-Agent Settings
Perfect-Information Games and Single-Agent Settings
Perfect-Information Games and Single-Agent Settings

Remaining game is too large
Perfect-Information Games
and Single-Agent Settings

Value substituted at leaf node $s$
is estimate of equilibrium value $\hat{\nu}(s)$

If estimate is perfect, local search
will find optimal policy (the equilibrium)
Depth-Limited Solving

Rock-Paper-Scissors+

Depth-Limited Rock-Paper-Scissors+
Depth-Limited Solving

Rock-Paper-Scissors+

Depth-Limited Rock-Paper-Scissors+
Depth-Limited Solving

Rock-Paper-Scissors+

Depth-Limited Rock-Paper-Scissors+
Depth-Limited Solving

Rock-Paper-Scissors+

- Naive approach: make state values a function of the state and the entire policy $\pi$
- Problem: extremely expensive

Depth-Limited Rock-Paper-Scissors+

- $P = 0.1$
- $P = 0.8$
- $P = 0.1$
Depth-Limited Solving

Rock-Paper-Scissors+

- Problem: still very expensive (DeepStack used 1.5 million core hours and could not beat prior top AIs).
- Problem: does not (currently) scale. In Texas Hold’em, input is ~2,000 floats. In five-card draw, input is ~5 billion floats. In Stratego, input is $> 10^{20}$. 

Depth-Limited Rock-Paper-Scissors+
Depth-Limited Solving in Modicum

[Brown et al. NIPS-18]

Rock-Paper-Scissors+

Depth-Limited Rock-Paper-Scissors+

- At leaf nodes, allow other player(s) one final action choosing between multiple policies for the remaining game
- Step 1: Solve subgame with current set of $P_2$ leaf-node policies
- Step 2: Calculate a $P_2$ best response
- Step 3: Add $P_2$ best response to set of leaf-node policies
- Repeat
Depth-Limited Solving in *Modicum*

[Brown et al. NIPS-18]

**Rock-Paper-Scissors+**

- At leaf nodes, allow other player(s) one final action choosing between multiple policies for the remaining game
- **Step 1:** Solve subgame with current set of $P_2$ leaf-node policies
- Step 2: Calculate a $P_2$ best response
- Step 3: Add $P_2$ best response to set of leaf-node policies
- Repeat

**Depth-Limited Rock-Paper-Scissors+**
Depth-Limited Solving in Modicum

[Brown et al. NIPS-18]

- At leaf nodes, allow other player(s) one final action choosing between multiple policies for the remaining game.
- Step 1: Solve subgame with current set of $P_2$ leaf-node policies.
- **Step 2: Calculate a $P_2$ best response.**
- Step 3: Add $P_2$ best response to set of leaf-node policies.
- Repeat.
Depth-Limited Solving in Modicum

[Brown et al. NIPS-18]

Rock-Paper-Scissors+

Depth-Limited Rock-Paper-Scissors+

- At leaf nodes, allow other player(s) one final action choosing between multiple policies for the remaining game.
- Step 1: Solve subgame with current set of $P_2$ leaf-node policies.
- Step 2: Calculate a $P_2$ best response.
- Step 3: Add $P_2$ best response to set of leaf-node policies.
- Repeat.
Depth-Limited Solving in Modicum

[Brown et al. NIPS-18]

**Rock-Paper-Scissors+**

1. At leaf nodes, allow other player(s) one final action choosing between multiple policies for the remaining game.
2. **Step 1**: Solve subgame with current set of $P_2$ leaf-node policies.
3. **Step 2**: Calculate a $P_2$ best response.
4. **Step 3**: Add $P_2$ best response to set of leaf-node policies.
5. Repeat.

**Depth-Limited Rock-Paper-Scissors+**
Depth-Limited Solving in *Modicum*

[Brown et al. NIPS-18]

Rock-Paper-Scissors+

Depth-Limited Rock-Paper-Scissors+

- At leaf nodes, allow other player(s) one final action choosing between multiple policies for the remaining game
- Step 1: Solve subgame with current set of $P_2$ leaf-node policies
- **Step 2: Calculate a $P_2$ best response**
- Step 3: Add $P_2$ best response to set of leaf-node policies
- Repeat
Depth-Limited Solving in *Modicum*

[Brown et al. NIPS-18]

**Rock-Paper-Scissors+**

- At leaf nodes, allow other player(s) one final action choosing between multiple policies for the remaining game
- Step 1: Solve subgame with current set of $P_2$ leaf-node policies
- Step 2: Calculate a $P_2$ best response
- **Step 3:** Add $P_2$ best response to set of leaf-node policies
- Repeat
Depth-Limited Solving in *Modicum*

[Brown et al. NIPS-18]

- $P_2$ decision is made at each leaf information set separately
  - 100 leaf infosets and 10 policies to choose from means $10^{100}$ choices

- For $P_1$ decisions below the depth limit, we assume $P_1$ plays according to the pre-computed approximate equilibrium

- The set of $P_2$ policies is pre-computed, not decided in real time
Exploitability Measurements

Exploitability of depth-limited solving in NLFH

Exploitability (mgb/s) vs Number of Values Per Leaf Nodes

- No Real-Time Reasoning
- Multi-Valued States

Graph showing how exploitability decreases as the number of values per leaf nodes increases.
Head-to-head performance of Modicum

- **Tartanian8** [2016 champion]
  - 2 million core hours
  - 18 TB of memory
  - No real-time reasoning

- **Slumbot** [2018 champion]
  - 250,000 core hours
  - 2 TB of memory
  - No real-time reasoning

- **Modicum**
  - 700 core hours
  - 16 GB of memory
  - Plays in real time with a 4-core CPU in 20 seconds per hand

<table>
<thead>
<tr>
<th></th>
<th>Tartanian8</th>
<th>Slumbot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modicum (no real-time reasoning)</td>
<td>-57 ± 13</td>
<td>-11 ± 8</td>
</tr>
<tr>
<td>Modicum (just one value per leaf node)</td>
<td>-10 ± 8</td>
<td>-1 ± 15</td>
</tr>
<tr>
<td>Modicum</td>
<td>6 ± 5</td>
<td>11 ± 9</td>
</tr>
</tbody>
</table>
Exploitability Measurements

Exploitability of depth-limited solving in NLFH

Exploitability (mbb/g) vs. Number of Values Per Leaf Nodes

- No Real-Time Reasoning
- Multi-Valued States
Head-to-head performance of Modicum

- **Tartanian8** [2016 champion]
  - 2 million core hours
  - 18 TB of memory
  - No real-time reasoning

- **Slumbot** [2018 champion]
  - 250,000 core hours
  - 2 TB of memory
  - No real-time reasoning

- **Modicum**
  - 700 core hours
  - 16 GB of memory
  - Plays in real time with a 4-core CPU in 20 seconds per hand

<table>
<thead>
<tr>
<th></th>
<th>Tartanian8</th>
<th>Slumbot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modicum (no real-time reasoning)</td>
<td>-57 ± 13</td>
<td>-11 ± 8</td>
</tr>
<tr>
<td>Modicum (just one value per leaf node)</td>
<td>-10 ± 8</td>
<td>-1 ± 15</td>
</tr>
<tr>
<td>Modicum</td>
<td>6 ± 5</td>
<td>11 ± 9</td>
</tr>
</tbody>
</table>
Head-to-head strength of top AIs

Stronger

Ours

Libratus (1/2017)

Modicum (5/2018)

Tartanian8 (12/2015) = ACPC 2016 winner

Tartanian7 (5/2014) = ACPC 2014 winner

Others’

Slumbot (12/2017) = ACPC 2018 winner

Slumbot (12/2015)

DeepStack (12/2016)
Key Takeaways

• In real-time planning, you must always consider how the opponent can adapt to changes in your policy
  – Except in perfect-information games

• Imperfect-information subgames cannot be solved in isolation

• States do not have a single well-defined value in imperfect-information games
Head-to-head strength of top AIs

- **Ours**
  - Libratus (1/2017)
  - Modicum (5/2018)
  - Tartanian8 (12/2015)
    - = ACPC 2016 winner
  - Tartanian7 (5/2014)
    - = ACPC 2014 winner

- **Others’**
  - Slumbot (12/2017)
    - = ACPC 2018 winner
  - Slumbot (12/2015)
  - DeepStack (12/2016)

Strength comparison:
- Libratus (1/2017) vs. Modicum (5/2018): 63 mbb/game vs. 6 mbb/game
Key Takeaways

• In real-time planning, you must always consider how the opponent can *adapt* to changes in your policy
  – Except in perfect-information games

• Imperfect-information subgames cannot be solved in isolation

• States do not have a single well-defined value in imperfect-information games
Other work

• How do we actually solve these games? Answer: CFR
  – Developed a form of CFR that is faster than the prior best by 3x

• Pruning in CFR (and Fictitious Play) [Brown & Sandholm NIPS-15, ICML-17]
  – Provably reduces computing and memory requirements
  – In practice, can speed up convergence by orders of magnitude

• Determining the optimal action(s) in a continuous action space
  [Brown & Sandholm AAAI-14]
Future Directions

• Bringing together techniques for perfect-information and imperfect-information games

• Semi-cooperative (general-sum) games, emergent communication

• Real-world applications: negotiations, security, auctions
Thank You!

Noam Brown
www.noambrown.com