Game Tree Search with Adaptation in Stochastic Imperfect Information Games

Darse BILLINGS, Aaron DAVIDSON, Terence SCHAUENBERG, Neil BURCH, Michael BOWLING, Robert HOLTE, Jonathan SCHAEFER, Duane SZAFRON

COMP 763 – Modern Computer Games
January 26, 2006
Outline

1. Introduction
2. History and Issues
3. Game Search Tree
4. Opponent Modeling
5. Experiments
6. Conclusion
Introduction

- Modeling the preferences and biases of humans is an important topic in AI

- We can easily gather enough data for a user, but using it to predict future patterns and behaviours is challenging
  - Even harder is to mine the data to predict human strategies in a competitive environment.
Introduction

- We use poker to explore challenging AI problems

- Why poker?
  - What distinguishes a good player from another is the ability to predict an opponent's hidden cards by his/her behaviour.
  - “Skillful opponent modeling is often the differentiating factor among world-class players.”
Introduction

- Current best program: PsOpti
  - Uses a minimax solution
  - Defensive strategy and assumes opponent has best cards.

  “You have a very strong program. Once you add opponent modeling to it, it will kill everyone.”
Introduction

5 cards
- High card
- Pair
- Two-Pairs
- Three of a kind
- Straight
- Flush
- Full House
- Four of a kind
- Straight Flush
Introduction

- **Texas Hold-'em**
  - Each player has 2 cards hidden from other players
  - Five *community* cards, which are shared among all players
  - *Call* (or *check*)
  - *Raise*
  - *Fold*

- Game ends when only one player left, or *showdown*. 
Introduction

- Why Texas Hold-'em?
  - Seen as “most strategically complex poker variant.”
  - It is used at the World Series of Poker to determine the champion.

- This paper concentrates on two-player limit Texas Hold-'em
Outline

1. Introduction
2. History and Issues
3. Game Search Tree
4. Opponent Modeling
5. Experiments
6. Conclusion
History and Issues

- 3 decent Poker A.I.'s: Loki, Poki, and PsOpti
  - Rule-Based Expert System
  - Simulations
  - Game-Theory
    - Nash equilibrium
History and Issues

- Nash Equilibrium
  - *Optimal Strategy*
  - Defensive, no risk
    - No player has an incentive to deviate from the strategy because the alternatives *could* lead to worst result.
  - Theoretically in long run, no player (human or computer) should be able to beat it.
Issues with Nash Equilibrium

- Impossible to compute a true Nash Equilibrium solution for Texas Hold-'em.
- It is a fixed strategy, and strong human players will be able to exploit its weaknesses.
- To defeat human players, it requires a program that observes opponents and adapts to dynamically changing conditions.
History and Issues

- Best is to use a maximal player
  - Exploit any biases or preferences
  - Takes risk if it believes to have a higher expected value (EV)
Outline

1. Introduction
2. History and Issues
3. Game Search Tree
4. Opponent Modeling
5. Experiments
6. Conclusion
Game Search Tree

- Expectimax
  - Similar to minimax search with the addition of chance nodes

- Example: Rolling a die
  - Sum all values of children weighted by the probability of the event occurring.
    - \( P(X = x) = \frac{1}{6} \)
Game Search Tree

- Expectimax
  - Cannot be used for poker
    - Imperfect information
    - Nodes of trees are not independent
    - Do not know probability function of a human player behaving a certain way for each event.
Game Search Tree

- **Miximax & Miximix**
  - EV is computed at each node using the information we know of the player.
  
  - *Miximax:* Mixed nodes for opponents max nodes for us.
  - Leads to predictable play
  
  - *Miximix:* Randomize our policy as well
Game Search Tree

- Issues:
  - How do we determine the relative probabilities for the opponent?
    - Look at past actions (i.e. same, or similar, betting sequence)
  - How do we calculate the EV of a leaf node?
    - Fold: Net amount won/loss
    - Showdown: PDF over strength of opponent's hand, using similar situations in the past.
Game Search Tree

- So we end up with 4 different type of nodes:
  - Chance
  - Opponent decision
  - Program decision
  - Leaf
Game Search Tree

- Chance Nodes
  - Weighted sum of the EV of subtree for each possible outcome
  - This is dependent on the cards each player holds (which cannot be calculated)

\[
EV(C) = \sum_{\forall i \in \text{outcomes}} P(C_i) \times EV(C_i)
\]
Game Search Tree

- Opponent Decision Nodes
  - Estimated probability of each branch (call, fold, raise)

$$EV(O) = \sum_{\forall i \in \{f, c, r\}} P(O_i) \times EV(O_i)$$
Game Search Tree

- Program Decision Nodes
  - If mixed policy, similar to $EV(O)$:
    
    $$EV(U) = \sum_{\forall i \in \{f, c, r\}} P(U_i) \times EV(U_i)$$

  - If we are maximizing $EV$ (maximax):
    
    $$EV(U) = \max(EV(U_f), EV(U_c), EV(U_r))$$
Game Search Tree

- **Leaf Nodes**
  - $L$: leaf node
  - $L_{pot}$: size of the pot
  - $L_{cost}$: cost of reaching leaf node
    - (in 2 player games, should be half of $L_{pot}$)
  - $P$ (win): Probability of winning

\[
    EV(L) = (P\text{ (win)} \times L_{pot}) - L_{cost}
\]
Game Search Tree
Outline

1. Introduction
2. History and Issues
3. Game Search Tree
4. Opponent Modeling
5. Experiments
6. Conclusion
Opponent Modeling

- Issues that make this problem difficult:
  - Must be rapid learning
    - Matches do not last thousands of hands
  - Strong players alternate their playing style
  - Only partial feedback
    - Often opponent cards are not revealed
      - Folding means what?
Opponent Modeling

- Unlike most Markov Decision Process problems, we are not looking at a static model.

- Handling observations:
  - Action decisions update betting frequencies corresponding to sequence of actions.
  - For showdowns, the hand rate (HR) shown by opponent is used to update the leaf node histogram.
Opponent Modeling

- $2 \times 9^4 = 13122$ leaf-level histograms
  - We don't have enough games to make sufficient number of observations to have reliable conclusion
  - Not to mention that worthy opponents usually change their strategies many times
    - We want to be able to base decisions on just dozens of hands rather than thousands
Opponent Modeling

- Generalize the observations
  - How do we accomplish this?
    - Finest level of granularity
      - Every sequence is distinct
    - Coarser abstraction:
      - Differentiate observations by number of bets and raises
      - Ignore at what stage if the hand they were made
Opponent Modeling

- Even coarser?
  - Sum total number of raises by both players
  - Ignore which player performed what action
  - Only 9 distinct classes!

- But remember:
  - More important to have usable data than have perfect correlations.
Opponent Modeling

- Their method is to use a mixture of all abstractions
  - All levels of abstraction contribute depending on how relevant the situation
Opponent Modeling

- Zero frequency problem
  - What happens when the program has no, or very little observations?
  - They combined a Nash equilibrium strategy to the mixing pot.
Opponent Modeling

- Players change their strategies often
  - We need to gradually forget old data and concentrate more on recent observations

- We use a history decay factor, $h$
  - We give all our observations a different weight depending on $h$.
  - Eg: $h = 0.95$
Outline

1. Introduction
2. History and Issues
3. Game Search Tree
4. Opponent Modeling
5. Experiments
6. Conclusion
Experiments

- Round-Robin Computer vs Computer
  - **Sparbot**: Latest version of PsOpti
    - Best program for this variant of poker
  - **Poki**: Formula based (w/opponent modelling)
    - Best program for 10-player poker
  - **Hobbybot**: Slowly adapting program
    - Designed to exploit Poki's flaws
  - **Jagbot**: Static formula-based
  - **Always Call**
  - **Always Raise**
Experiments

- Each match consisted of at least 10,000 hands
- Standard deviation: ±0.03 \( \text{sb/hand} \)
- Vs. Sparbot, Vexbot needed thousands of hands before able to exploit Sparbot's flaws

<table>
<thead>
<tr>
<th>Program</th>
<th>Vexbot</th>
<th>Sparbot</th>
<th>Hobbot</th>
<th>Poki</th>
<th>Jagbot</th>
<th>A.Call</th>
<th>A.Raise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vexbot</td>
<td>+0.052</td>
<td>+0.349</td>
<td>+0.601</td>
<td>+0.477</td>
<td>+1.042</td>
<td>+2.983</td>
<td></td>
</tr>
<tr>
<td>Sparbot</td>
<td>-0.052</td>
<td>+0.033</td>
<td>+0.093</td>
<td>+0.059</td>
<td>+0.474</td>
<td>+1.354</td>
<td></td>
</tr>
<tr>
<td>Hobbybot</td>
<td>-0.349</td>
<td>-0.033</td>
<td>+0.287</td>
<td>+0.099</td>
<td>+0.044</td>
<td>+0.463</td>
<td></td>
</tr>
<tr>
<td>Poki</td>
<td>-0.601</td>
<td>-0.093</td>
<td>-0.287</td>
<td>+0.149</td>
<td>+0.510</td>
<td>+2.139</td>
<td></td>
</tr>
<tr>
<td>Jagbot</td>
<td>-0.477</td>
<td>-0.059</td>
<td>-0.099</td>
<td>-0.149</td>
<td>+0.597</td>
<td>+1.599</td>
<td></td>
</tr>
<tr>
<td>Always Call</td>
<td>-1.042</td>
<td>-0.474</td>
<td>-0.044</td>
<td>-0.510</td>
<td>-0.597</td>
<td>=0.000</td>
<td></td>
</tr>
<tr>
<td>Always Raise</td>
<td>-2.983</td>
<td>-1.354</td>
<td>-0.463</td>
<td>-2.139</td>
<td>-1.599</td>
<td>=0.000</td>
<td></td>
</tr>
</tbody>
</table>
Experiments

- Vexbot vs Humans
  - A lot less hands played
  - Very competitive vs. experts
  - Results showed a consistent marked increased in win rate after 200-400 hands
    - Due to opponent-specific modeling?

<table>
<thead>
<tr>
<th>Num</th>
<th>Rating</th>
<th>sb/h</th>
<th>Hands</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert</td>
<td>-0.022</td>
<td>3739</td>
</tr>
<tr>
<td>2</td>
<td>Intermediate</td>
<td>+0.136</td>
<td>1507</td>
</tr>
<tr>
<td>3</td>
<td>Intermediate</td>
<td>+0.440</td>
<td>821</td>
</tr>
<tr>
<td>4</td>
<td>Intermediate</td>
<td>+0.371</td>
<td>773</td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. History and Issues
3. Game Search Tree
4. Opponent Modeling
5. Experiments
6. Conclusion
Conclusion

- Following contributions:
  - Miximax & Miximix
  - Using opponent modeling to refine EV
  - Abstraction for compression of large set of observable data
  - Vexbot
    - Best poker program
    - Competitive vs expert humans.
Conclusion

- Future work:
  - Not take as long to learn new opponent
  - Improving the abstractions
  - Generalize to games with > 2 players
Conclusion

Questions