Monte-Carlo tree search for multi-player, no-limit Texas hold'em poker

Guy Van den Broeck
Deceptive play

Should I bluff?
Opponent modeling

Should I bluff?

Is he bluffing?
Game of chance

Should I bluff?

Who has the Ace?

What are the odds?

Is he bluffing?
Exploitation

Should I bluff?

Who has the Ace?

What are the odds?

Is he bluffing?

I'll bet because he always calls
Huge state space

Should I bluff?

Who has the Ace?

What are the odds?

Is he bluffing?

What can happen next?

I'll bet because he always calls
Risk management & Continuous action space

Should I bet $5 or $10?

Should I bluff?

Who has the Ace?

What are the odds?

Is he bluffing?

What can happen next?

I'll bet because he always calls
Take-Away Message:
We can solve all these problems!
Problem Statement

- A bot for Texas hold'em poker
  - No-Limit & > 2 players
    - Not done before!
  - Exploitative, not game theoretic
    - Game tree search + Opponent modeling

- Applies to any problem with either
  - incomplete information
  - non-determinism
  - continuous actions
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion
## Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
  - Opponent model

- Conclusion
Minimax trees: deterministic

Tic-tac-toe, checkers, chess, go,…
Poker Game Tree

- Minimax trees: deterministic
  - Tic-tac-toe, checkers, chess, go, ...

- Expecti(mini)max trees: chance
  - Backgammon, ...

\[
\begin{array}{ccc}
\text{max} & \text{min} \\
\end{array}
\]
Poker Game Tree

Minimax trees: deterministic
- Tic-tac-toe, checkers, chess, go, ...

Expecti(mini)max trees: chance
- Backgammon, ...

Miximax trees: hidden information
- + opponent model
my action

fold

call

raise

♠️  ♥️  ♣️  ♦️
my action

- fold
- call
- raise

Resolve
my action

0 fold

Resolve
call
raise

Playing cards: spades, hearts, diamonds, clubs
my action

- 0 fold
- call
- raise

Resolve

Reveal Cards

0.5

...
my action

Resolve

0 fold

call

raise

Reveal Cards

0.5

0.5

-1

... 3

-1 3
my action

0 fold

Resolve

1 call

raise

Reveal Cards

0.5 0.5

-1 3
my action

Resolve

Reveal Cards

0.5 0.5

-1...3

0.6 fold

op-1 action

0.3 call

0.1 raise
my action

Resolve

Reveal Cards

opp-1 action

opp-2 action

raise

call

fold

fold

fold

call

raise

-1

0.5

0.5

3

-1

0.6

4

0.3

0.1

...
my action

Resolve

Reveal Cards

opp-1 action

opp-2 action

0.6 fold

0.3 call

0.1 raise

fold

call

raise

fold

...
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion
Short Experiment
Opponent Model

- Set of probability trees
- Weka's M5'
- Separate model for
  - Actions
    \[ P(A_i | A_0 \ldots A_{i-1}, C_0 \ldots C_i) \]
  - Hand cards at showdown
    \[ P(H | A_0 \ldots A_n, C_0 \ldots C_n) \]
Fold Probability

nbAllPlayer Raises <= 1.5 :
|    callFrequency <= 0.128 :
|     |    nbActionsThisRound <= 2.5 :
|     |     |    potOdds <= 0.28 :
|     |     |     |    AF <= 2.585 : 0.6904
|     |     |     |    AF > 2.585 :
|     |     |     |     |    potSize <= 3.388 :
|     |     |     |     |     |    round=flop <= 0.5 : 0.8068
|     |     |     |     |     |     |    round=flop > 0.5 : 0.6896
|     |     |     |     |     |     |    potSize > 3.388 : 0.8198
|     |     |     |    potOdds > 0.28 :
|     |     |     |     |    stackSize <= 97.238 :
|     |     |     |     |     |    callFrequency <= 0.038 : 0.8838
|     |     |     |     |     |     |    callFrequency > 0.038 :
|     |     |     |     |     |     |     |    round=flop <= 0.5 : 0.8316
|     |     |     |     |     |     |     |    round=flop > 0.5 :
|     |     |     |     |     |     |     |     |    nbSeatedPlayers <= 7.5 : 0.6614
|     |     |     |     |     |     |     |    nbSeatedPlayers > 7.5 : 0.7793
|     |     |     |     |     |     |    stackSize > 97.238 :
|     |     |     |     |     |     |     |    potSize <= 4.125 :
|     |     |     |     |     |     |     |     |    foldFrequency <= 0.813 : 0.7839
|     |     |     |     |     |     |     |     |    foldFrequency > 0.813 : 0.9037
|     |     |     |     |     |     |     |    potSize > 4.125 : 0.8623
|     |     |    nbActionsThisRound > 2.5 :
|     |     |     |    potOdds <= 0.218 :
|     |     |     |     |    callFrequency <= 0.067 : 0.8753
|     |     |     |     |     |    callFrequency > 0.067 : 0.7661
|     |     |     |     |    potOdds > 0.218 :
|     |     |     |     |     |    AF <= 2.654 : 0.8818
|     |     |     |     |     |     |    AF > 2.654 : 0.921
(Can also be relational)

Tilde probability tree [Ponsen08]

- active_player(X), has_position(X)
  - yes
  - previous_action(X, raise)
    - yes
    - [bet: 0.65; call: 0.25; fold: 0.10]
    - no
    - [bet: 0.10; call: 0.15; fold: 0.75]
  - no
    - [bet: 0.25; call: 0.55; fold: 0.20]

- playing_style(X, tight)
- pot_odds(low)
- ...
Opponent Ranks

⚠️  Learn distribution of hand ranks at showdown

![Bar chart showing the probability of different hand ranks across various number of raises.](chart.png)
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
  - Opponent model

- Conclusion
Traversing the tree

- Limit Texas Hold’em
  - $10^{18}$ nodes
  - Fully traversable
- No-limit
  - $>10^{71}$ nodes
  - Too large to traverse
  - Sampled, not searched
  - Monte-Carlo Tree Search
Monte-Carlo Tree Search

[Chaslot08]
Selection

In each node:

\[ \hat{V}(P) \] is an estimate of the reward \( r(P) \)
\[ T(P) \] is the number of samples
Selection

In each node:

\[ \hat{V}(P) \]
\[ \hat{V}(c_i) + C \sqrt{\frac{\ln T(P)}{T(c_i)}} \]

is an estimate of the reward \( r(P) \)

is the number of samples

\( \text{UCT (Multi-Armed Bandit)} \)
Selection

In each node:

\[
\hat{V}(P) \quad \text{is an estimate of the reward} \quad r(P)
\]

\[
T(P) \quad \text{is the number of samples}
\]

\[
\hat{V}(c_i) + C \sqrt{\frac{\ln T(P)}{T(c_i)}}
\]


\textbf{UCT (Multi-Armed Bandit)}

exploitation
Selection

In each node:

\[ \hat{V}(P) \] is an estimate of the reward \( r(P) \)

\[ T(P) \] is the number of samples

\[ \hat{V}(c_i) + C \sqrt{\frac{\ln T(P)}{T(c_i)}} \]

**UCT (Multi-Armed Bandit)**

- **exploitation**
- **exploration**
Selection

In each node:

\[ \hat{V}(P) \] is an estimate of the reward \( r(P) \)

\[ T(P) \] is the number of samples

\[ \hat{V}(c_i) + C \sqrt{\frac{\ln T(P)}{T(c_i)}} \]

- **UCT (Multi-Armed Bandit)**
  - **exploitation**
  - **exploration**

\[ P(c_i) \sim \exp \left( -2.4 \frac{\hat{V}(c_{best}) - \hat{V}(c_i)}{\sqrt{2(\sigma(c_{best})^2 + \sigma(c_i)^2)}} \right) \]

**CrazyStone**
Expansion
Simulation
Backpropagation

\[
\hat{V}(P) \quad \text{is an estimate of the reward} \quad r(P)
\]
\[
T(P) \quad \text{is the number of samples}
\]
Backpropagation

\[ \hat{V}(P) \text{ is an estimate of the reward } r(P) \]
\[ T(P) \text{ is the number of samples} \]

Sample-weighted average

\[ \hat{V}(n) = \sum_j \frac{T(c_j)}{T(n)} \hat{V}(c_j) \]
Backpropagation

\[ \hat{V}(P) \] is an estimate of the reward \( r(P) \)
\[ T(P) \] is the number of samples

- **Sample-weighted average**

\[
\hat{V}(n) = \sum_j \frac{T(c_j)}{T(n)} \hat{V}(c_j)
\]

- **Maximum child**

\[
\hat{V}(P) = \max_j \hat{V}(c_j)
\]
Initial experiments

- 1*MCTS + 2*rule based
- Exploitative!
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
  - Opponent model

- Conclusion
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift

- Conclusion
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift

- Conclusion
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift

- Conclusion
MCTS for games with uncertainty?

- Expected reward distributions (ERD)
- Sample selection using ERD
- Backpropagation of ERD

[VandenBroeck09]
Expected reward distribution

MiniMax

Estimating $r(P)$

10 samples

100 samples

$\infty$ samples

Variance
Expected reward distribution

Estimating $r(P)$

10 samples

100 samples

$\infty$ samples

Variance
Expected reward distribution

Estimating $r(P)$

- 10 samples
- 100 samples
- $\infty$ samples

Variance
Expected reward distribution

Estimating $r(P)$

- 10 samples
- 100 samples
- $\infty$ samples

Variance
Expected reward distribution

Estimating $r(P)$

10 samples

100 samples

$\infty$ samples

Variance Sampling
Expected reward distribution

Estimating $r(P)$

- 10 samples
- 100 samples
- $\infty$ samples

- Variance
- Sampling

MiniMax

ExpectiMax/MixiMax

$\infty$ samples

Variance

Sampling
Expected reward distribution

Estimating

MiniMax $r(P)$

10 samples

100 samples

$\infty$ samples

ExpectiMax/MixiMax $r(P)$

Variance Sampling
Expected reward distribution

**MiniMax**
\[ r(P) \]

**ExpectiMax/MixiMax**
\[ r(P) \]

- Estimating
- 10 samples
- 100 samples
- \( \infty \) samples

**Variance**

**Sampling**
Expected reward distribution

- Estimating
  - MiniMax $r(P)$
  - ExpectiMax/MixiMax $r(P)$

- Variance
- Sampling

- 10 samples
- 100 samples
- $\infty$ samples
Expected reward distribution

100 samples

MiniMax
\[ r(P) \]

ExpectiMax/MixiMax
\[ r(P) \]

Estimating

10 samples

100 samples

\[ \infty \] samples

Variance

Sampling

Uncertainty + Sampling
Expected reward distribution

- **MiniMax**
  \[ r(P) \]
  - Estimating
  - 10 samples
  - 100 samples
  - \( \infty \) samples
  - Variance
  - Sampling

- **ExpectiMax/MixiMax**
  \[ r(P) \]
  - \( E[r(P)] / T(P) \)
  - 10 samples
  - 100 samples
  - \( \infty \) samples
  - Uncertainty + Sampling
Expected reward distribution

Estimating

MiniMax $r(P)$

ExpectiMax/MixiMax $r(P)$

ExpectiMax/MixiMax $\mathbb{E}[r(P)]$

Variance

Sampling

Uncertainty + Sampling
Expected reward distribution

Estimating $r(P)$

- 10 samples
- 100 samples
- $\infty$ samples

Variance

Sampling

Uncertainty + Sampling

MiniMax

ExpectiMax/MixiMax

ExpectiMax/MixiMax $\mathbb{E}[r(P)]$
Expected reward distribution

Estimating $r(P)$

10 samples

100 samples

$\infty$ samples

Variance Sampling Uncertainty + Sampling

MiniMax ExpectiMax/MixiMax ExpectiMax/MixiMax

$\frac{E[r(P)]}{T(P)}$
Expected reward distribution

Estimating $r(P)$

10 samples

100 samples

$\infty$ samples

Variance

Sampling

Uncertainty + Sampling

Sampling
Expected reward distribution

Estimating $r(P)$

MiniMax

ExpectedMax/MixiMax

$\mathbb{E}[r(P)]$

/ $T(P)$

10 samples

100 samples

$\infty$ samples

Variance

Sampling

Uncertainty $+$ Sampling

Sampling 😊
ERD selection strategy

Objective?
- Find maximum expected reward
- Sample more in subtrees with
  (1) High expected reward
  (2) Uncertain estimate

UCT does (1) but not really (2)
CrazyStone does (1) and (2) for deterministic games (Go)

UCT+ selection: $\hat{V}(c_i) + C.\sigma_{\hat{V},c_i}$
ERD selection strategy

Objective?
- Find maximum expected reward
- Sample more in subtrees with
  1. High expected reward
  2. Uncertain estimate

UCT does (1) but not really (2)
CrazyStone does (1) and (2) for deterministic games (Go)

UCT+ selection: \( \hat{V}(c_i) + C \cdot \sigma_{V,c_i} \)

“Expected value under perfect play”
ERD selection strategy

Objective?
- Find maximum expected reward
- Sample more in subtrees with
  1. High expected reward
  2. Uncertain estimate

UCT does (1) but not really (2)
CrazyStone does (1) and (2) for deterministic games (Go)

UCT+ selection: $\hat{V}(c_i) + C \cdot \sigma_{\hat{V}, c_i}$

“Measure of uncertainty due to sampling”
ERD max-distribution
backpropagation

\[ \hat{V}(P) \]

max

\[ \hat{V}(c_i) \]

3

4
ERD max-distribution backpropagation

\[ \hat{V}(P) \]

\[ \text{max} \]

\[ \hat{V}(c_i) \]

sample-weighted

3.5

3 4
ERD max-distribution
backpropagation

\[ \hat{V}(P) \]

max

A

B

\[ \hat{V}(c_i) \]

3

4

3.5

4

sample-weighted

max
When the game reaches P, we'll have more time to find the real \( E[r(P)] \).
ERD max-distribution
backpropagation

\[ \hat{V}(P) \]

\[
\begin{align*}
\hat{V}(c_i) & \\
3 & \quad 4 & \quad 4.5 \\
A & \quad \text{max} & \quad \text{max-distribution} \\
B & \quad \text{sample-weighted} & \\
\ldots & \quad \ldots & \\
\end{align*}
\]
ERD max-distribution

backpropagation

\[ \hat{V}(P) \]

max

\[ \hat{V}(c_i) \]

\[ \begin{array}{c|c|c}
   & A<4 & A>4 \\
\hline
B<4 & 0.8*0.5 & 0.2*0.5 \\
B>4 & 0.8*0.5 & 0.2*0.5 \\
\end{array} \]

\[ P(B<4) = 0.5 \quad P(B>4) = 0.5 \]
\[ P(A<4) = 0.8 \quad P(A>4) = 0.2 \]

\[ P(\text{max}(A,B)>4) = 0.6 \]
\[ > 0.5 \]

3

4

4.5
Experiments

- 2*MCTS
  - Max-distribution
  - Sample-weighted

- 2*MCTS
  - UCT+ (stddev)
  - UCT

![Graph 1](image1.png)

![Graph 2](image2.png)
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift

- Conclusion
Dealing with continuous actions

- Sample discrete actions

- Progressive unpruning [Chaslot08] (ignores smoothness of EV function)

- ... 

- Tree learning search (work in progress)
Based on regression tree induction from *data streams*
- training examples arrive *quickly*
- nodes *split* when significant reduction in stddev
- training examples are immediately *forgotten*

Edges in TLS tree are not actions, but *sets of actions*, e.g., (raise in [2,40]), (fold or call)

MCTS provides a *stream* of (action,EV) examples

Split action sets to reduce stddev of EV (when significant)
Tree learning search

max

Bet in \([0,10]\)  
\{Fold, Call\}
Tree learning search

Bet in $[0,10]$  
\{Fold, Call\}

Expected value as a function of the bet
Tree learning search

\[ \text{max} \]

Bet in \([0,10]\)

\{Fold, Call\}

\[ \text{max} \]

Optimal split at 4
Tree learning search

- \( \max \) on \([0, 10]\)
- \{Fold, Call\}
- \( \max \) on \([0, 4]\)
- Bet in \([4, 10]\)

Tree learning search

P1

P2

P3

one action of P1

one action of P2
Selection Phase

Each node has EV estimate, which generalizes over actions
Expansion

P1

Selected Node

P2
Expansion

Expanded node

Represents any action of P3
Backpropagation

New sample; Split becomes significant
Backpropagation

New sample; Split becomes significant
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift

- Conclusion
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift

- Conclusion
Online learning of opponent model

- Start from (safe) model of general opponent
- Exploit weaknesses of specific opponent

Start to learn model of specific opponent

(Exploration of opponent behavior)
Multi-agent interaction
Multi-agent interaction

Yellow learns model for Blue and changes strategy.
Multi-agent interaction

Yellow learns model for Blue and changes strategy

Yellow doesn't profit!
Multi-agent interaction

Yellow learns model for Blue and changes strategy.

Yellow doesn't profit!

Green profits without changing strategy!!
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search

- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
      - Concept drift

- Conclusion
While learning from a stream, the training examples in the stream change.

In opponent model: changing strategy

"Changing gears is not just about bluffing, it's about changing strategy to achieve a goal."

Learning with concept drift

adapt quickly to changes

yet robust to noise

(recognize recurrent concepts)
Basic approach to concept drift

- Maintain a window of training examples
  - large enough to learn
  - small enough to adapt quickly
  - without 'old' concepts
- Heuristics to adjust window size
  - based on FLORA2 framework [Widmer92]
4 components of a single opponent model

Accuracy

Start online learning

Concept drift

Window size
Bad parameters for heuristic

Accuracy

Window size

CAUTION
NOT
ROBUST
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion
Conclusions

- First exploitive poker bot for
  - *No-limit* Holdem
  - > 2 players

- Apply in other games
  - backgammon
  - computational pool
  - ...

- Challenge for **MCTS**
  - games with uncertainty
  - continuous action space

- Challenge for **ML**
  - online learning
  - concept drift
  - (relational learning)
Thanks for listening!