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

Guy Van den Broeck
Deceptive play

Should I bluff?
Should I bluff?

Is he bluffing?

Opponent modeling
Should I bluff?

Who has the Ace?

Is he bluffing?

Incomplete information
Game of chance

Should I bluff?

Who has the Ace?

Is he bluffing?

What are the odds?
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

Huge state space
Risk management &
Continuous action space

Should I bluff?

Should I bet $5 or $10?

Is he bluffing?

Who has the Ace?

What are the odds?

What can happen next?

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

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
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,…
\textbf{Poker Game Tree}

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

- Expecti(mini)max trees: chance
  - Backgammon, …
Poker Game Tree

- Minimax trees: deterministic
  - Tic-tac-toe, checkers, chess, go,…
  - Max \(\max\)  Min \(\min\)

- Expecti(mini)max trees: chance
  - Backgammon, …
  - Max \(\max\)  Min \(\min\)  Mix \(\text{mix}\)

- Miximax trees: hidden information
  - Max \(\max\)  Mix \(\text{mix}\)  Mix \(\text{mix}\)  + opponent model
my action

- fold
- call
- raise

Resolve

[Spade] [Heart] [Diamond] [Club]
my action

0 fold

Resolve

call

raise

Reveal Cards

0.5 0.5

...
my action

- 0 fold
- call
- raise

Resolve

Reveal Cards

- 0.5
- 0.5

-1 3
my action

- 0 fold
- 1 call
- raise

Resolve

Reveal Cards

- 0.5
- 0.5
- -1
- 3
my action

Resolve

Reveal Cards

opp-1 action

opp-2 action

0.5

0.5

-1

3

0.6

0.3

0.1

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                     no
previous_action(X, raise) [bet:0.65; call:0.25; fold:0.10]

yes                     no
[bet:0.10; call:0.15; fold:0.75] [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

Number of Raises

Rank Bucket

Probability

Opponent Ranks

Number of Raises
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

Repeated X times

[Chaslot08]
In each node:

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

In each node:

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

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

¶ UCT (Multi-Armed Bandit)

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

In each node:

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

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

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

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

In each node:

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

⚠️ UCT (Multi-Armed Bandit)

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

exploitation

exploration
Selection

In each node:

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

$T(P)$ is the number of samples

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

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

- UCT (Multi-Armed Bandit)
- CrazyStone
Expansion Simulation
Backpropagation

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

$T(P)$ is the number of samples
Backpropagation

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

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

- Sample-weighted average

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

Estimating $r(P)$

10 samples

100 samples

∞ samples

Variance
Expected reward distribution

Estimating $r(P)$

10 samples

100 samples

∞ 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

MiniMax
\[ r(P) \]

ExpectiMax/MixiMax
\[ r(P) \]

10 samples

100 samples

∞ samples

Variance Sampling
Expected reward distribution

- Estimating $r(P)$
  - 10 samples
  - 100 samples
  - $\infty$ samples

- Variance
- Sampling

MiniMax

ExpectiMax/MixiMax
Expected reward distribution

Estimating

$\text{MiniMax } r(P)$

$\text{ExpectiMax/MixiMax } r(P)$

10 samples

100 samples

$\infty$ samples

Variance

Sampling
Expected reward distribution

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

- Variance
  - $r(P)$

- Sampling
  - ExpectiMax/MixiMax
  - $r(P)$

MiniMax

Variance

Sampling
Expected reward distribution

Estimating $r(P)$

10 samples

100 samples

∞ samples

Variance Sampling Uncertainty + Sampling
Expected reward distribution

- MiniMax
  
  \( r(P) \)

- ExpectiMax/MixiMax
  
  \( r(P) \) / \( T(P) \)

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

Estimating

10 samples

100 samples

\( \infty \) samples

Variance

Sampling

Uncertainty + Sampling
Expected reward distribution

Estimating

\[ r(P) \]

MiniMax

ExpectiMax/MixiMax

\[ E[r(P)] \]

/ \( T(P) \)

10 samples

100 samples

\( \infty \) samples

Variance

Sampling

Uncertainty + Sampling
Expected reward distribution

100 samples

MiniMax

Estimating

ExpectiMax/MixiMax

10 samples

100 samples

∞ samples

Variance

Sampling

Uncertainty + Sampling

ExpectiMax/MixiMax

\[ r(P) \]

\[ \frac{1}{T(P)} \]

\[ E[r(P)] \]
Expected reward distribution

Estimating $r(P)$

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

Variance: Sampling

Uncertainty + Sampling
Expected reward distribution

Estimating $r(P)$

10 samples

100 samples

∞ samples

Variance  Sampling  Uncertainty + Sampling  Sampling
Expected reward distribution

- **MiniMax**: $r(P)$
- **ExpectiMax/MixiMax**: $r(P)$
- **ExpectiMax/MixiMax**: $E[r(P)]$

Estimating $r(P)$

- 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: \( \hat{V}(c_i) + C \cdot \sigma_{\hat{V},c_i} \)
  1. (1)
  2. (2)
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: $\hat{V}(c_i) + C \cdot \sigma_{\hat{V},c_i}$

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

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

backpropagation

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

\[ \max \]

A

B

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

3

4
ERD max-distribution ♠️ ♥️ ♦️ ♣️
backpropagation

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

max

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

sample-weighted

3.5

3

4
ERD max-distribution
backpropagation

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

\[ \max \]

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

sample-weighted

3.5

\[ \max \]

3

4
ERD max-distribution
backpropagation

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

max

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

3

sample-weighted

3.5

max

4

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

backpropagation
ERD max-distribution

backpropagation

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

\[ \max \]

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

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

\[ \max \]

\[ \text{max}(A,B) > 4 \] \( \Rightarrow 0.5 \)

\[ \text{max}(A,B) \leq 4 \] \( \Rightarrow 0.6 \)

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

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

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

\[ > 0.5 \]

\[ 3 \]

\[ 4 \]

\[ 4.5 \]
Experiments

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

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

---

![Graph showing computation time vs. average profit for different bots.](image_url)
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)
Tree learning search

- 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]  \{\text{Fold, Call}\}

max

Expected value as a function of the bet
Tree learning search

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

Optimal split at 4
Tree learning search ♠️ ♥️ ♦️ ♣️

```
max
Bet in [0,10]  {Fold, Call}
```

```
max
Bet in [0,4]  Bet in [4,10]
```

```
max
max
```

```
?
?
?
?```
Tree learning search

one action of P1

one action of P2
Selection Phase

Each node has EV estimate, which generalizes over actions.
Expansion

Selected Node
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!

Green profits without changing strategy!!

Yellow doesn't profit!
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!