Monte Carlo Tree Search for Simultaneous Move Games

Marc Lanctot

Joint work with Mark Winands, Viliam Lisý, Christopher Wittlinger, Mandy Tak

1 Department of Knowledge Engineering, Maastricht University
2 Dept. of Computer Science and Engineering, FEE, Czech Technical University in Prague

December 2nd, 2013
Talk Overview

- Introduction and Background
  - Monte Carlo Search is Everywhere
  - Monte Carlo Tree Search (MCTS)
  - Consistency in Tree Search

- Simultaneous Move Games and MCTS

- Experiments and Results

- Conclusion and Future work
Talk Overview

- Introduction and Background
  - Monte Carlo Search is Everywhere
  - Monte Carlo Tree Search (MCTS)
  - Consistency in Tree Search

- Simultaneous Move Games and MCTS

- Experiments and Results

- Conclusion and Future work

  → incl. some new stuff, too!
Monte Carlo Search is Everywhere

Coulom ’06, Kocsis & Szepesvári ’06, Chaslot et al.’07, Gelly et al.’07, Winands & Björnsson ’08, more..

\[ p(x_0) = a_0 + x_0(a_1 + x_0(a_2 + \cdots + x_0(a_n - 1 + b_0 x_0) \cdots)) \]
\[ = a_0 + x_0(a_1 + x_0(a_2 + \cdots + x_0(b_n - 1) \cdots)) \]
\[ ; \]
\[ = a_0 + x_0(b_n) \]
\[ = b_0. \]

Kuipers et al.’12

Van Eyck ’13, Zhu ’13

Lavalle & Kuffner ’01, Perez et al.’11

Cazenave ’05, Van den Broek ’09, Silver & Veness ’10, Ponsen et al.’11, Cowling et al.’12, Amato et al.’13

Couetoux et al.’11
Monte Carlo Tree Search: Overview

The selection function is applied recursively until a leaf node is reached. One or more nodes are created. One simulated game is played. The result of this game is backpropagated in the tree. Repeated X times.

(Coulom ’06), (Kocsis & Szepesvári ’06), (Chaslot et al.’07), (Browne et al.’12)
Monte Carlo Tree Search (MCTS) Example
Monte Carlo Tree Search (MCTS) Example
Monte Carlo Tree Search (MCTS) Example
Monte Carlo Tree Search (MCTS) Example
Monte Carlo Tree Search (MCTS) Example

```
3 −2 0 1 4 −1 −3 1 2 0 −1 5 3 2 −3 0
```

- visits = 2
- rsum = 2

- visits = 1
- rsum = 1

- visits = 4
- rsum = 3

- visits = 1
- rsum = 2

Max
Min
Max
Min
Monte Carlo Tree Search (MCTS) Example

visits = 1
rsum = 1

visits = 2
rsum = 2

visits = 1
rsum = 1

visits = 1
rsum = −1

visits = 2
rsum = 1

visits = 5
rsum = 2

Max

Min

Max

Min

3 −2 1 0 4 −1 −3 1 2 0 −1 5 3 2 −3 0
Monte Carlo Tree Search (MCTS) Example
Monte Carlo Tree Search (MCTS) Example
Monte Carlo Tree Search (MCTS) Example

visits = 8
rsum = 0

visits = 3
rsum = 1

visits = 1
rsum = 1

visits = 1
rsum = −1

visits = 1
rsum = −1

visits = 1
rsum = −1

visits = 4
rsum = −2

visits = 2
rsum = −1

visits = 1
rsum = 2

visits = 1
rsum = −1

visits = 1
rsum = −1

visits = 1
rsum = −1

visits = 3
rsum = 1

visits = 8
rsum = 0

Max

Max

Max

Min

Min

Min
Bandit-Based Selection

To select a node, employ bandit algorithms (i.e. UCB (Auer et al.’02)).
Bandit-Based Selection

To select a node, employ bandit algorithms (i.e. UCB (Auer et al.’02)).

$\Rightarrow$ UCT algorithm (Kocsis & Szepesvári ’06).
Algorithm $A(s, t)$ computes strategy $\sigma(s)$ at state $s$ after $t$ time units.
Algorithm $A(s, t)$ computes strategy $\sigma(s)$ at state $s$ after $t$ time units.

In two-player zero-sum games, for every state $s$, $\exists$ Nash equilibrium (optimal) strategy $\sigma^*(s)$. 
Algorithm $A(s, t)$ computes strategy $\sigma(s)$ at state $s$ after $t$ time units.

In two-player zero-sum games, for every state $s$, $\exists$ Nash equilibrium (optimal) strategy $\sigma^*(s)$.

Definition (Weak/Asymptotic Consistency)
A search algorithm is (weakly) consistent if $\forall s, \lim_{t \to \infty} A(s, t) = \sigma^*(s)$. 

Computation (search) time...
Consistency in MCTS

Is MCTS consistent?

- Yes, in sequential MP perfect information, using max $n$ (Sturtevant et al. '08)

MCTS-Solver

- Converges faster
- Increases play strength

See: (Winands et al. ’08, Baier ’13, Nijssen ’13, more..)
Consistency in MCTS

Is MCTs consistent?

- Yes!
Is MCTS consistent?

- Yes!

  → in sequential 2P games with perfect information, using UCT
Consistency in MCTS

Is MCTS consistent?

- Yes!
  → in sequential 2P games with perfect information, using UCT

- > 2 players?
  Yes, in seq. MP perfect information, using max" (Sturtevant et. al. ’08)
Is MCTS consistent?

- Yes!
  → *in sequential 2P games with perfect information, using UCT*

- > 2 players?
  Yes, in seq. MP perfect information, using max” (Sturtevant et. al. ’08)

- MCTS-Solver
  ▶ converges faster
  ▶ increases play strength
  ▶ See: (Winands et al.’08, Baier ’13, Nijssen ’13, more..)

Many optimizations that work better than plain UCT in practice!
Simultaneous Move Games

Each player takes a single action (simultaneously) and game over!
Multiple Stages: Goofspiel

- $(13!)^3 \approx 2.41 \cdot 10^{29}$ unique sequences
- Backward induction, proposed (Ross 1971)
- Solved (Rhoads & Bartholdi 2012)
Let $\sigma_{col} = (r, p, s)$. Maximize $V$ such that $r, p, s \geq 0$ and

\[
\begin{align*}
\frac{1}{2}r + s & \geq V \\
r + \frac{1}{2}p & \geq V \\
p + \frac{1}{2}s & \geq V \\
r + p + s & = 0
\end{align*}
\]
Multi-stage: Simultaneous Move Games
Backward Induction

1 0 0 1 3 4 1 2 1 1 0 0

1 0 0 1 3 4 1 2 1 1 0 0

13 / 35
Backward Induction
Backward Induction
Simultaneous Move Minimax

Extension to classic game-tree search: “SM-Minimax”:

- Depth-first search, as before
- Values sent back up are solutions to LPs
- Finite depth? → Evaluation function
- Weakly consistent

Recent work:

- Simultaneous Move $\alpha\beta$ (SMAB), (Saffidine et al.’12)
- Move Pruning in Serialized $\alpha\beta$, (Bosansky et al.’13)
Simultaneous Move MCTS

Extension to MCTS: “SM-MCTS”:
Simultaneous Move MCTS

Extension to MCTS: “SM-MCTS”:

- Selection policy chooses a joint action \((a_{row}, a_{col})\)
Simultaneous Move MCTS

Extension to MCTS: “SM-MCTS”:
- Selection policy chooses a joint action \((a_{\text{row}}, a_{\text{col}})\)
- Payoff matrices contain (changing) estimates
Simultaneous Move MCTS

Extension to MCTS: “SM-MCTS”:

- Selection policy chooses a joint action \((a_{\text{row}}, a_{\text{col}})\)
- Payoff matrices contain (changing) estimates
- Regret minimization in unknown matrix games:

  \[ \text{Stochastic Bandits (UCB)} \rightarrow \text{“Adversarial” Bandits (Exp3, RM)} \]
Simultaneous Move MCTS

Extension to MCTS: “SM-MCTS”:

- Selection policy chooses a joint action \((a_{row}, a_{col})\)
- Payoff matrices contain (changing) estimates
- Regret minimization in unknown matrix games:
  
  Stochastic Bandits (UCB) → “Adversarial” Bandits (Exp3, RM)

- Consistency?
Simultaneous Move MCTS

Extension to MCTS: “SM-MCTS”:
- Selection policy chooses a joint action \((a_{row}, a_{col})\)
- Payoff matrices contain (changing) estimates
- Regret minimization in unknown matrix games:
  - *Stochastic Bandits (UCB) → “Adversarial” Bandits (Exp3, RM)*
- Consistency?

Previous work:
- Backward induction (Ross 1971, Buro ’03)
- Sequential UCT (Several 2008 - 2011)
- Decoupled UCT (DUCT) (Finnsson et. al. ’08, Perick et. al. ’11)
  - Provably inconsistent (Shafiei et. al. ’09)
- Pruning in SM-MCTS (Finnsson ’12)
- Exp3 at each stage (Auger ’11, Teytaud & Flory ’11)
  - Both: empirical evidence of consistency
Our SM-MCTS Variants

So far, we have looked at:

- Decoupled UCT
- Sequential UCT
- UCB1-Tuned (Sequential and Decoupled)
- Exp3 (Auer et al.’95)
- Regret Matching (RM) (Hart & Mas-Colell ’00)
- Online Outcome Sampling (OOS)
- $\epsilon$-minmax

in several domains ...
Sequential UCT

Model the game as a sequential game.

Then apply usual UCT for selection.
Decoupled UCT (DUCT)

Each player $i$ keeps their own reward estimates and visits for $a_i \in A_i(s)$

Use UCB for selection. When done simulations:

- **DUCT(max)**: choose $\arg\max_{a \in A(s)} X_a / v_a$
- **DUCT(mix)**: choose $a$ with prob $v_a / \sum_{b \in A_i(s)} v_b$
In (Shafiei et al. ’09),

- DUCT(mix) converges to \( \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \right) \) in R, P, S
- In biased R, P, S:

\[
\begin{array}{ccc}
R & 0.5 & 0.25 & 1 \\
P & 0.75 & 0.5 & 0.45 \\
S & 0 & 0.55 & 0.5 \\
\end{array}
\]

- DUCT converges to a cycle
- However, strategy is not a Nash eq.

- Similar no-convergence in Kuhn Poker (Ponsen et al. ’11)
Each player $i$ keeps their own reward estimates for $a_i \in A_i(s)$

Let $p(a) = \exp(X_a) / \sum_{b \in A_i(s)} \exp(X_b)$

- Selection: sample $a$ using $\hat{p}(a) = \gamma / |A_i(s)| + (1 - \gamma)p(a)$
- Obtain reward $r$ from below, update $X_a \leftarrow X_a + r / \hat{p}(a)$
- After sims, choose according to empirical freq. of samples
Regret Matching (RM)

Each player $i$ keeps their own **cumulative regret** for $a_i \in A_i(s)$

Define $x^+ = \max(0, x)$, and $p(a) = r_a^+ / \sum_{b \in A_i(s)} r_b^+$

- Selection: sample $a$ using $\hat{p}(a) = \gamma / |A_i(s)| + (1 - \gamma)p(a)$
- Obtain reward $r$, accumulate regret for not playing $b \neq a$,
- Add current strategies $p(b)$ to average strategy table $s_b$ for each $b$.
- After simulations, choose by normalizing $s_a$ for $a \in A_i(s)$. 

![Diagram](image-url)
$\epsilon$-minmax

After many simulations, we have an approximation for each $X_{a_1a_2}$.

- Construct and solve LP to get mixed $\sigma_1(s), \sigma_2(s)$.
- Then, each player selects $a_i \sim \epsilon \cdot \text{Unif}(A_i(s)) + (1 - \epsilon) \cdot \sigma_i(s)$
- MCTS version of MinimaxQ (Littman ’98)
Online Outcome Sampling (OOS)

- Counterfactual regret minimization (Zinkevich et al. '08)

  ▶ Approaches Nash eq. in imperfect information setting
  ▶ Intended for offline use
  ▶ Used to compute Poker strategies

Monte Carlo CFR

→ MCTS version of outcome sampling MCCFR

Use regret matching over sampled counterfactual regrets

Converges to Nash equilibrium over time!
Online Outcome Sampling (OOS)

- Counterfactual regret minimization (Zinkevich et al. '08)
  - Approaches Nash eq. in imperfect information setting
  - Intended for offline use
  - Used to compute Poker strategies
Online Outcome Sampling (OOS)

- **Counterfactual regret minimization** (Zinkevich et al. '08)
  - Approaches Nash eq. in imperfect information setting
  - Intended for offline use
  - Used to compute Poker strategies

- **Monte Carlo CFR** (Lanctot et al. '09)
Online Outcome Sampling (OOS)

- Counterfactual regret minimization (Zinkevich et al. '08)
  - Approaches Nash eq. in imperfect information setting
  - Intended for offline use
  - Used to compute Poker strategies
- Monte Carlo CFR (Lanctot et al. '09)
- Online Outcome Sampling
  → MCTS version of outcome sampling MCCFR
Online Outcome Sampling (OOS)

- Counterfactual regret minimization (Zinkevich et al. '08)
  - Approaches Nash eq. in imperfect information setting
  - Intended for offline use
  - Used to compute Poker strategies

- Monte Carlo CFR (Lanctot et al. '09)

- Online Outcome Sampling
  - MCTS version of outcome sampling MCCFR

- Use regret matching over sampled counterfactual regrets
Online Outcome Sampling (OOS)

- Counterfactual regret minimization (Zinkevich et al. '08)
  - Approaches Nash eq. in imperfect information setting
  - Intended for offline use
  - Used to compute Poker strategies

- Monte Carlo CFR (Lanctot et al. '09)

- Online Outcome Sampling
  - MCTS version of outcome sampling MCCFR

- Use regret matching over sampled counterfactual regrets

*Converges to Nash equilibrium over time!*
Win/Loss Goofspiel(13) Performance

<table>
<thead>
<tr>
<th>P1  \ P2</th>
<th>RND</th>
<th>DUCT(max)</th>
<th>DUCT(mix)</th>
<th>Exp3</th>
<th>OOS</th>
<th>OOS+</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUCT(max)</td>
<td>76.0</td>
<td>78.3</td>
<td>57.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUCT(mix)</td>
<td>78.3</td>
<td>57.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp3</td>
<td></td>
<td></td>
<td></td>
<td>80.0</td>
<td>55.8</td>
<td>48.4</td>
</tr>
<tr>
<td>OOS</td>
<td></td>
<td></td>
<td></td>
<td>73.1</td>
<td>55.3</td>
<td>43.8</td>
</tr>
<tr>
<td>OOS+</td>
<td></td>
<td></td>
<td></td>
<td>77.7</td>
<td>67.0</td>
<td>53.3</td>
</tr>
<tr>
<td>RM</td>
<td></td>
<td></td>
<td></td>
<td>80.9</td>
<td>63.3</td>
<td>53.2</td>
</tr>
</tbody>
</table>

Number refers to % win rate of row player type.
## Point Difference Goofspiel(11) Exploitability

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mean $E_{x_2}$</th>
<th>Mean $E_{x_4}$</th>
<th>Mean simulations per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUCT(max)</td>
<td>7.43 ± 0.15</td>
<td>12.87 ± 0.13</td>
<td>124127 ± 286</td>
</tr>
<tr>
<td>DUCT(mix)</td>
<td>5.10 ± 0.05</td>
<td>7.96 ± 0.02</td>
<td>124227 ± 286</td>
</tr>
<tr>
<td>Exp3</td>
<td>5.77 ± 0.10</td>
<td>10.12 ± 0.08</td>
<td>125165 ± 61</td>
</tr>
<tr>
<td>OOS</td>
<td>4.02 ± 0.06</td>
<td>7.92 ± 0.04</td>
<td>186962 ± 361</td>
</tr>
<tr>
<td>OOS+</td>
<td>5.59 ± 0.09</td>
<td>9.30 ± 0.08</td>
<td>85940 ± 200</td>
</tr>
<tr>
<td>RM</td>
<td>5.56 ± 0.10</td>
<td>9.36 ± 0.07</td>
<td>138284 ± 249</td>
</tr>
</tbody>
</table>

- Lower $E_{x_d}$ = closer to Nash eq.
WL-Goof(4) and PD-Goof(4) Full Exploitability

- **Mean Exploitability in WL-Goof(4)**
  - X-axis: Time
  - Y-axis: Distance to Nash eq. (lower = closer)
  - Legend: DUCT, Exp3, OOS+, RM

- **Mean Exploitability in PD-Goof(4)**
  - X-axis: Time
  - Y-axis: Distance to Nash eq. (lower = closer)
  - Legend: DUCT, Exp3, OOS+, RM

- x-axis is time, y-axis is distance to Nash eq. (lower = closer)
New Results I: Tron

(Lanctot et al.'13). Based on Bachelor thesis of Christopher Wittlinger.

Try to survive and block opponent in a maze.
New Results I: Tron

Three boards below, plus empty board.

Heuristic knowledge in playouts (space estimation + predictive expansion strategies; see also Bachelor thesis of Niek Den Teuling.)
New Results I: Tron

<table>
<thead>
<tr>
<th>Variant</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUCB1T(max)</td>
<td>65%</td>
<td>62%</td>
<td>62%</td>
<td>59%</td>
<td>62.32 ± 0.56%</td>
</tr>
<tr>
<td>DUCB1T(mix)</td>
<td>56%</td>
<td>57%</td>
<td>53%</td>
<td>53%</td>
<td>54.82 ± 0.61%</td>
</tr>
<tr>
<td>UCB1T</td>
<td>58%</td>
<td>57%</td>
<td>49%</td>
<td>54%</td>
<td>54.32 ± 0.55%</td>
</tr>
<tr>
<td>RM</td>
<td>56%</td>
<td>52%</td>
<td>51%</td>
<td>53%</td>
<td>53.13 ± 0.62%</td>
</tr>
<tr>
<td>UCT</td>
<td>47%</td>
<td>54%</td>
<td>55%</td>
<td>49%</td>
<td>51.39 ± 0.55%</td>
</tr>
<tr>
<td>DUCT(max)</td>
<td>43%</td>
<td>54%</td>
<td>49%</td>
<td>50%</td>
<td>49.05 ± 0.61%</td>
</tr>
<tr>
<td>DUCT(mix)</td>
<td>39%</td>
<td>40%</td>
<td>43%</td>
<td>36%</td>
<td>39.51 ± 0.64%</td>
</tr>
<tr>
<td>Exp3</td>
<td>35%</td>
<td>24%</td>
<td>38%</td>
<td>45%</td>
<td>35.47 ± 0.61%</td>
</tr>
</tbody>
</table>

- Percentage is a win rate
- Results of the different variants played against each other
- ± refers to 95% confidence intervals.
(Lisý et al.’13) shows that:

- *Any* regret-minimizing alg. leads to weak consistency in SM-MCTS
- Must back-propagate the means for guarantee
- Worst-case analysis of Exp3 and RM on random games
### Preliminary Results I

Win percentages of $\epsilon$-minmax in Goofspiel:

<table>
<thead>
<tr>
<th>vs.</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUCT(max)</td>
<td>75.70 %</td>
</tr>
<tr>
<td>DUCT(mix)</td>
<td>48.60 %</td>
</tr>
<tr>
<td>Exp3</td>
<td>78.50 %</td>
</tr>
<tr>
<td>OOS</td>
<td>31.05 %</td>
</tr>
<tr>
<td>OOS$^+$</td>
<td>55.75 %</td>
</tr>
<tr>
<td>RM</td>
<td>47.55 %</td>
</tr>
</tbody>
</table>

- All are roughly $\pm 3.0$ for 95% c.i.
- Must solve LP only so often
- Must decay $\epsilon$
- Seems more sensitive to parameters
In Tron, run RM with purification: if probability < 0.2, flush it to 0 and renormalize.

<table>
<thead>
<tr>
<th></th>
<th>RM</th>
<th>RM + purification</th>
</tr>
</thead>
<tbody>
<tr>
<td>vs. DUCB1T(max)</td>
<td>31.52 %</td>
<td>35.07 %</td>
</tr>
<tr>
<td>vs. DUCT(max)</td>
<td>68.48 %</td>
<td>50.30 %</td>
</tr>
</tbody>
</table>

- RM(purification) wins 54.44% againsts RM
- All are roughly ±2.0 for 95% c.i.
Preliminary Results III: Oshi-Zumo

Solved in (Buro ’03). All results are versus DUCT(max)

<table>
<thead>
<tr>
<th></th>
<th>OZ [50,3,1]</th>
<th>OZ [15,3,1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUCT(mix)</td>
<td>16.60 %</td>
<td>36.30 %</td>
</tr>
<tr>
<td>Exp3</td>
<td>31.10 %</td>
<td>44.95 %</td>
</tr>
<tr>
<td>OOS</td>
<td>11.40 %</td>
<td>25.50 %</td>
</tr>
<tr>
<td>OOS⁺</td>
<td>23.85 %</td>
<td>42.40 %</td>
</tr>
<tr>
<td>RM</td>
<td>41.80 %</td>
<td>58.60 %</td>
</tr>
</tbody>
</table>

All are roughly ±2.5 for 95% c.i.
Conclusions

In Goofspiel:
- Regret matching and OOS$^+$ perform best in Goofspiel
- DUCT(max) performs worst overall
- DUCT(mix) surprisingly good
- OOS converges to Nash in the limit

In Tron:
- DUCB1T(max) is the clear best
Conclusions

In Goofspiel:
- Regret matching and $OOS^+$ perform best in Goofspiel
- DUCT(max) performs worst overall
- DUCT(mix) surprisingly good
- OOS converges to Nash in the limit

In Tron:
- DUCB1T(max) is the clear best

Future work:
- Apply in general game-playing (with Mandy Tak)
- Adaptive algorithms
- Compare to SM-MCTS move pruning (Finnsson ’12)
- Compare to SMAB (Saffidine et al. 2012)
- Compare to Serialized Alpha-Beta (Bosansky et al. 2013)
- Extend to fully imperfect information setting