Monte Carlo Tree Search for Simultaneous Move Games

Marc Lanctot[1]

Joint work with Mark Winands[1], Viliam Lisý[2], Christopher Wittlinger[1], Mandy Tak[1]

[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

$\rightarrow$ incl. some new stuff, too!

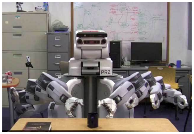
Monte Carlo Search is Everywhere

Coulom ’06, Kocsis & Szepesvári ’06,

Chaslot et al.’07, Gelly et al.’07,

Winands & Björnsson ’08, more..

Van Eyck ’13, Zhu ’13



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

$$
\begin{aligned}
p(x_0) &= a_0 + x_0(a_1 + x_0(a_2 + \cdots + x_0(a_{n-1} + b_n x_0)\cdots)) \\
&= a_0 + x_0(a_1 + x_0(a_2 + \cdots + x_0(b_{n-1})\cdots)) \\
&\vdots \\
&= a_0 + x_0(b_1) \\
&= b_0.
\end{aligned}
$$

Kuipers et al.’12

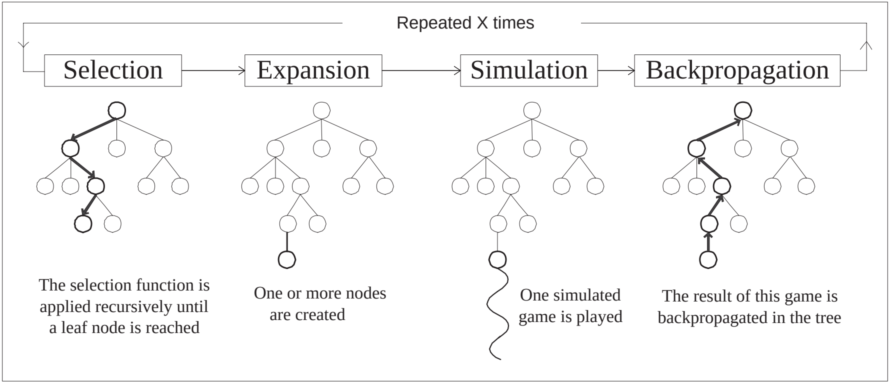
Cazenave ’05, Van den Broek ’09,

Silver & Veness ’10, Ponsen et al.’11,

Cowling et al.’12, Amato et al.’13

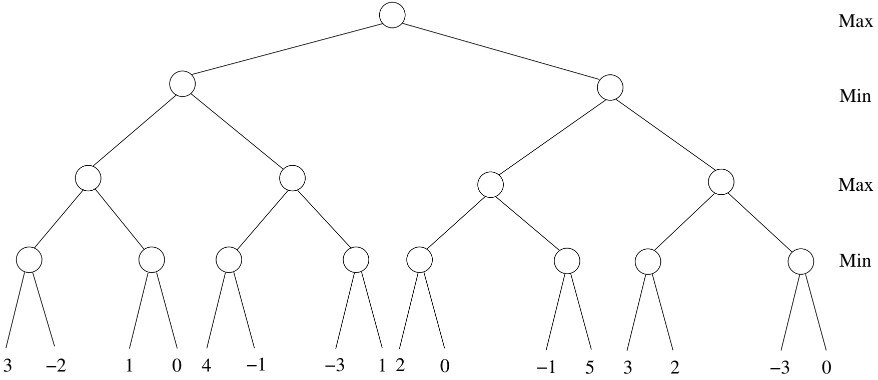
Couetoux et al.’11

Monte Carlo Tree Search: Overview

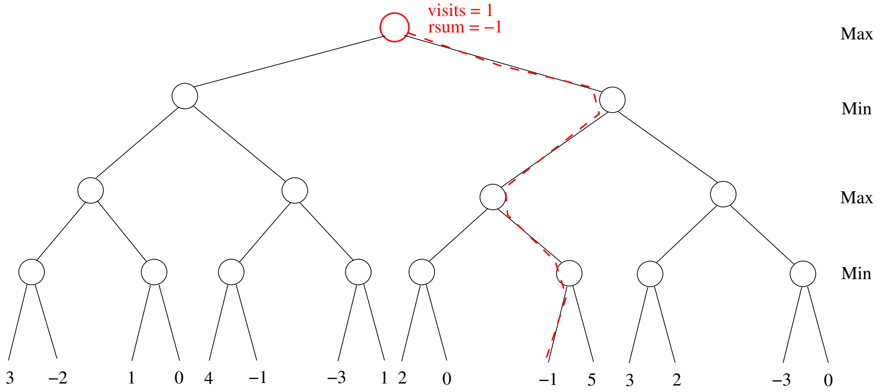


(Coulom ’06), (Kocsis & Szepesvári ’06), (Chaslot et al.’07), (Browne et al.’12)

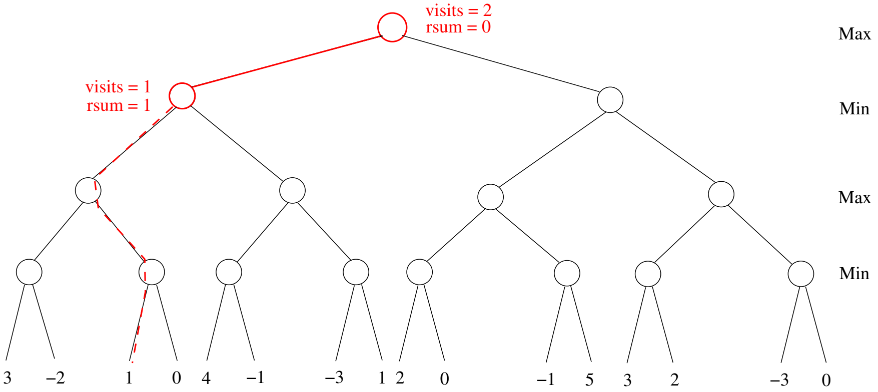
Monte Carlo Tree Search (MCTS) Example



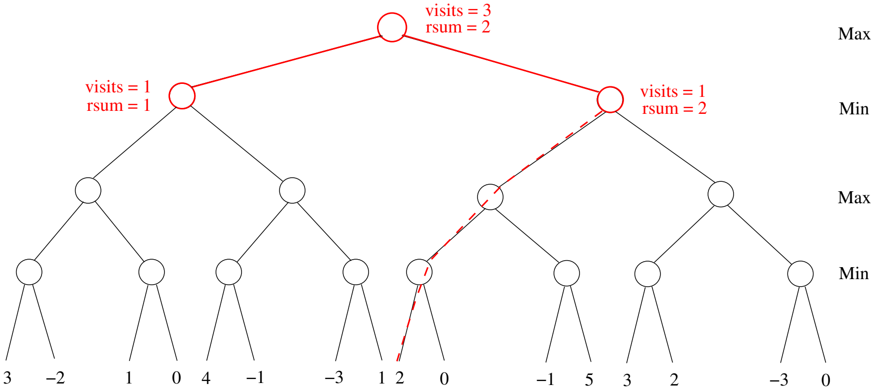
Monte Carlo Tree Search (MCTS) Example



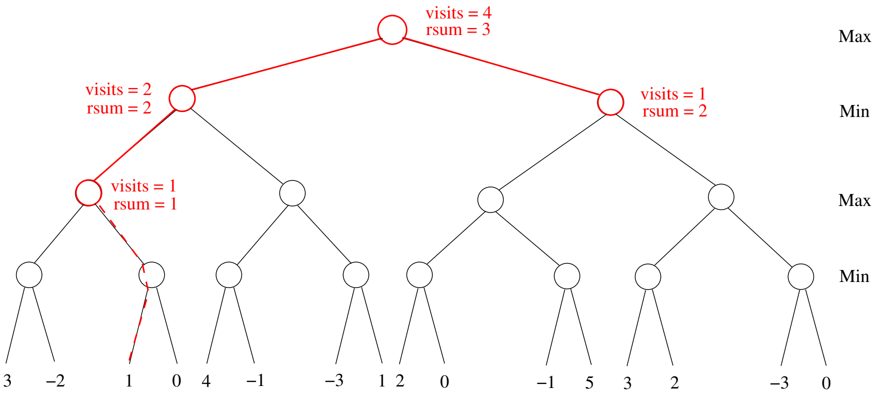
Monte Carlo Tree Search (MCTS) Example



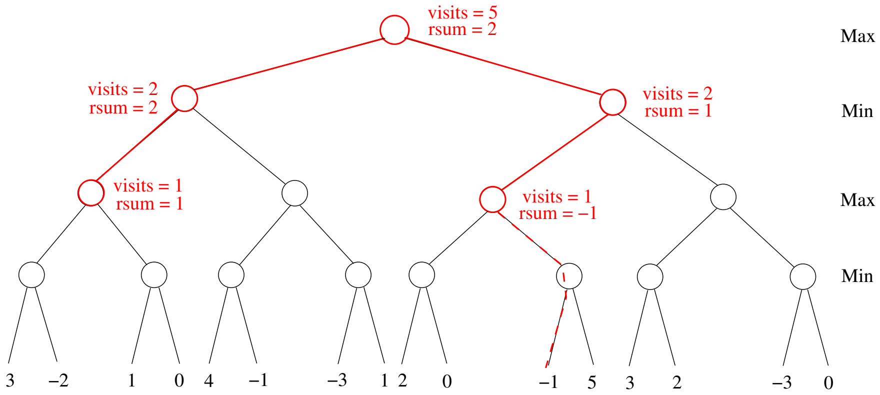
Monte Carlo Tree Search (MCTS) Example



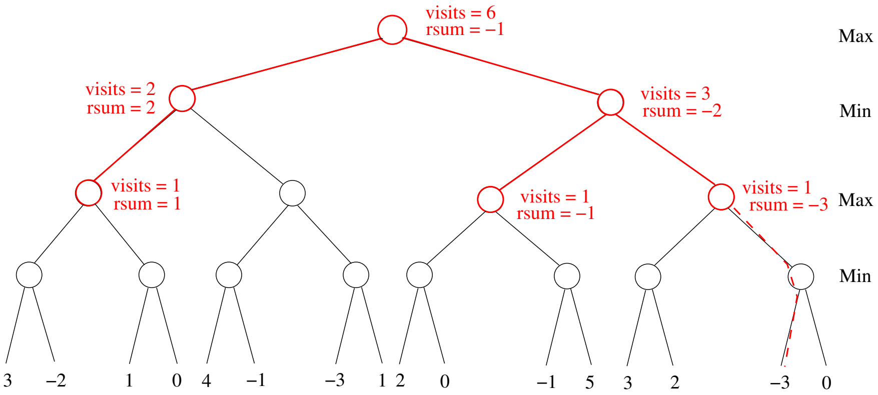
Monte Carlo Tree Search (MCTS) Example



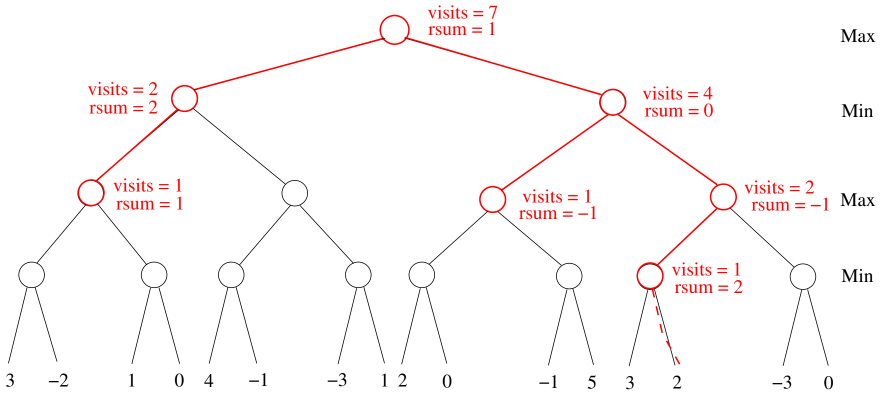
Monte Carlo Tree Search (MCTS) Example



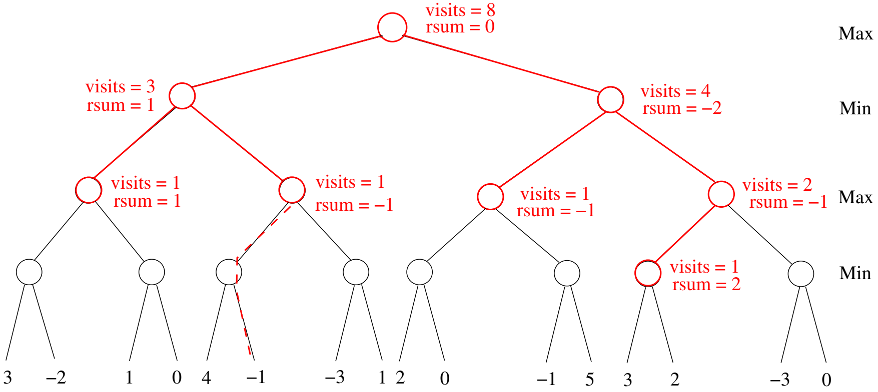
Monte Carlo Tree Search (MCTS) Example



Monte Carlo Tree Search (MCTS) Example

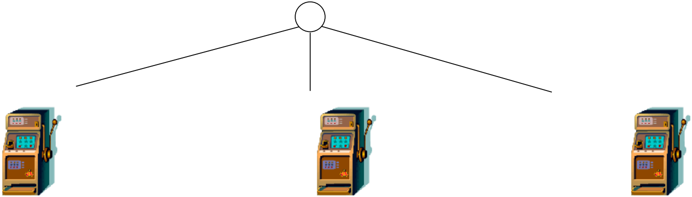


Monte Carlo Tree Search (MCTS) Example



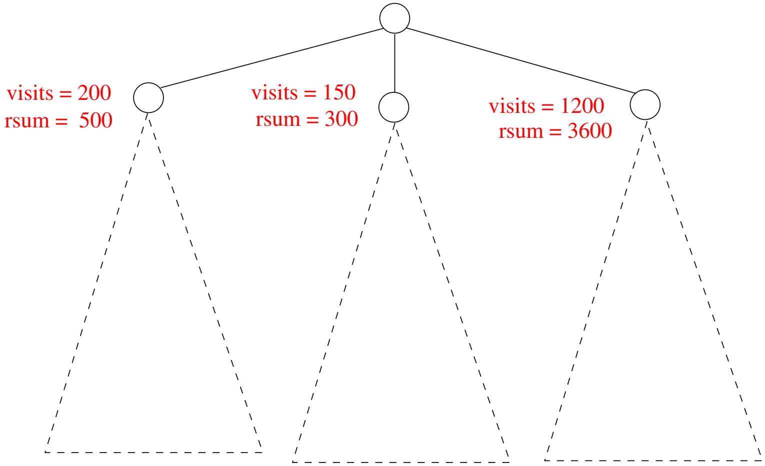
Bandit-Based Selection

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)).



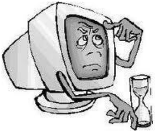
$\Rightarrow$ UCT algorithm (Kocsis & Szepesvári '06).

Consistency in Game Tree Search

Algorithm $A(s,t)$ computes strategy $\sigma(s)$ at state s after t time units.

Computation(search) time...

start computation | - - - - - - t seconds - - - - - - | output strategy

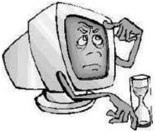


Consistency in Game Tree Search

Algorithm $A(s, t)$ computes strategy $\sigma(s)$ at state s after t time units.

Computation(search) time...

start computation | - - - - - - - - - - t seconds - - - - - - - - - - | output strategy



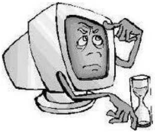
In two-player zero-sum games, for every state s, $\exists$ Nash equilibrium (optimal) strategy $\sigma^*(s)$.

Consistency in Game Tree Search

Algorithm $A(s,t)$ computes strategy $\sigma(s)$ at state s after t time units.

Computation(search) time...

start computation | - - - - - - - - - - t seconds - - - - - - - - - - | output strategy



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

Definition (Weak/Asymptotic Consistency)

A search algortihm is *(weakly) consistent* if $\forall s, \lim_{t\to\infty} A(s,t) = \sigma^*(s)$.

Consistency in MCTS

Is MCTS consistent?

Consistency in MCTS

Is MCTS consistent?

- Yes!

Consistency in MCTS

Is MCTS consistent?

- Yes!
  $\rightarrow$ *in sequential 2P games with perfect information, using UCT*

Consistency in MCTS

Is MCTS consistent?

- Yes!
  $\rightarrow$ *in sequential 2P games with perfect information, using UCT*
- > 2 players?
  Yes, in seq. MP perfect information, using max^n (Sturtevant et. al. '08)

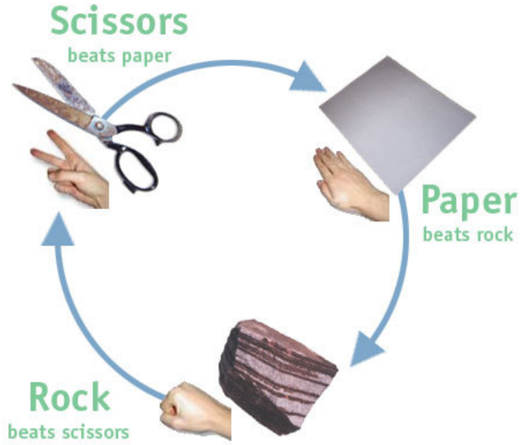
Consistency in MCTS

Is MCTS consistent?

- Yes!
  $\rightarrow$ *in sequential 2P games with perfect information, using UCT*
- > 2 players?
  Yes, in seq. 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..)

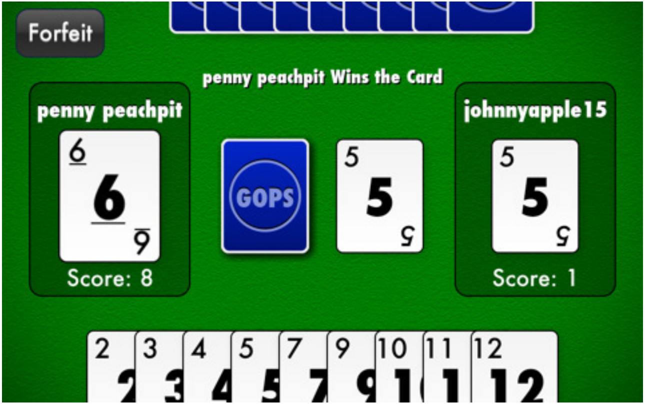
Many optimizations that work better than plain UCT in practice!

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)

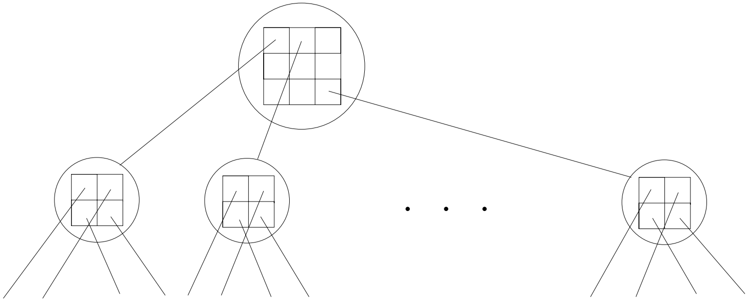
Single Stage: Matrix Game

	r	p	s
R	0.5	0	1
P	1	0.5	0
S	0	1	0.5

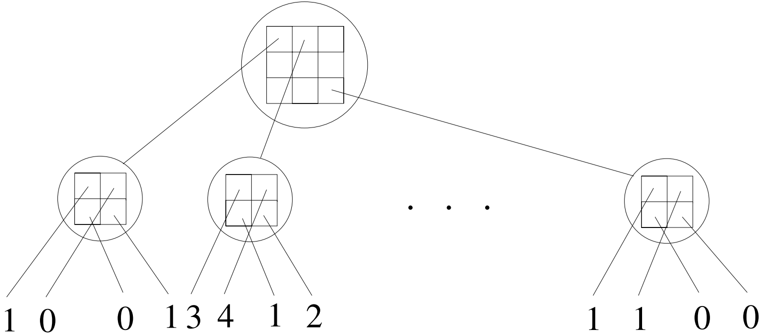
Let $\sigma_{col} = (r, p, s)$. Maximize V such that $r, p, s \geq 0$ and

$$\begin{array}{ccccccc} \frac{1}{2}r & & & + & s & \geq & V \\ r & + & \frac{1}{2}p & & & \geq & V \\ & & p & + & \frac{1}{2}s & \geq & V \\ r & + & p & + & s & = & 0 \end{array}$$

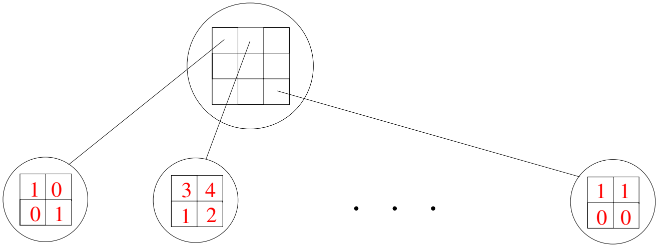
Multi-stage: Simultaneous Move Games



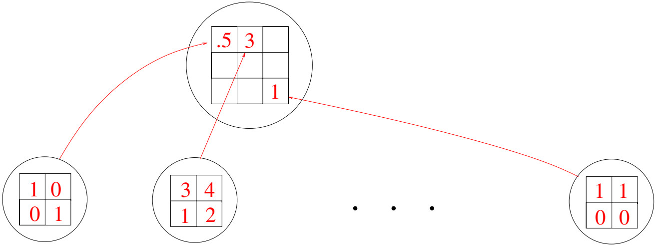
Backward Induction



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? $\rightarrow$ 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})

Extension to MCTS: “SM-MCTS”:

- Selection policy chooses a joint action (a_{row}, a_{col})
- Payoff matrices contain (changing) estimates

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) $\rightarrow$ *“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) $\rightarrow$ *"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) $\rightarrow$ *"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 consitency

Our SM-MCTS Variants

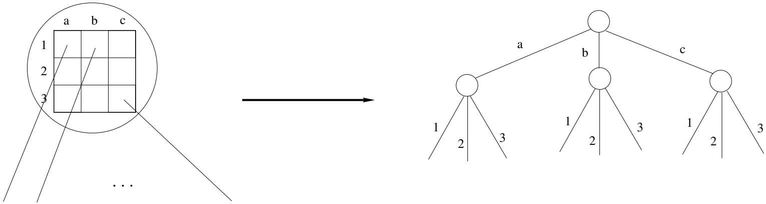
So far, we have looked at:

- 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)
- ϵ-minmax

in several domains ...

Sequential UCT

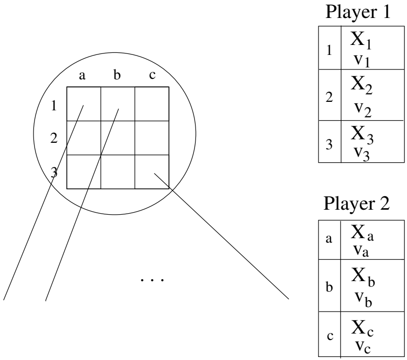
Model the game as a sequential game.



Then apply usual UCT for selection.

Decoupled UCT (DUCT)

Each player i keeps their own reward estimates and visits for $a_i \in \mathcal{A}_i(s)$



Use UCB for selection. When done simulations:

- DUCT(max): choose $\operatorname{argmax}_{a \in \mathcal{A}(s)} X_a / v_a$
- DUCT(mix): choose a with prob $v_a / \sum_{b \in \mathcal{A}_i(s)} v_b$

DUCT Consistency?

In (Shafiei et al. '09),

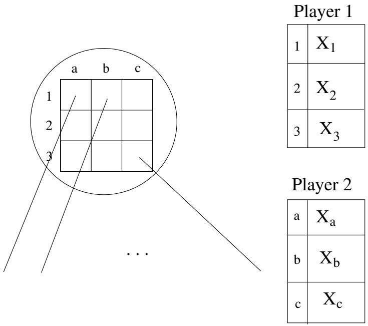
- DUCT(mix) converges to $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ in R, P, S
- In biased R, P, S:

	r	p	s
R	0.5	0.25	1
P	0.75	0.5	0.45
S	0	0.55	0.5

  - DUCT converges to a cycle
  - However, strategy is not a Nash eq.
- Similar no-convergence in Kuhn Poker (Ponsen et al. '11)

Exp3

Each player i keeps their own reward estimates for $a_i \in \mathcal{A}_i(s)$

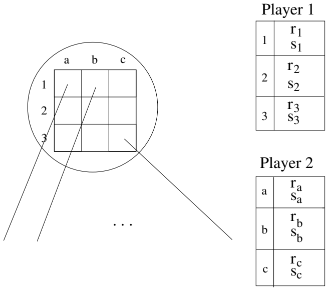


Let $p(a) = \exp(X_a) / \sum_{b \in \mathcal{A}_i(s)} \exp(X_b)$

- Selection: sample a using $\hat{p}(a) = \gamma / |\mathcal{A}_i(s)| + (1 - \gamma) p(a)$
- 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 $a_i \in \mathcal{A}_i(s)$



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

- Selection: sample a using $\hat{p}(a) = \gamma / |\mathcal{A}_i(s)| + (1 - \gamma) p(a)$
- 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 $a \in \mathcal{A}_i(s)$.

ϵ-minmax

After many simulations, we have an approximation for each $X_{a_1 a_2}$.

	a	b	c
1	$\overline{\mathrm{X}}_{1a}$	$\overline{\mathrm{X}}_{1b}$	$\overline{\mathrm{X}}_{1c}$
2	$\overline{\mathrm{X}}_{2a}$	$\overline{\mathrm{X}}_{2b}$	$\overline{\mathrm{X}}_{2c}$
3	$\overline{\mathrm{X}}_{3a}$	$\overline{\mathrm{X}}_{3b}$	$\overline{\mathrm{X}}_{3c}$

- Construct and solve LP to get mixed $\sigma_1(s), \sigma_2(s)$.
- Then, each player selects $a_i \sim \epsilon \cdot \text{Unif}(\mathcal{A}_i(s)) + (1-\epsilon) \cdot \sigma_i(s)$
- MCTS version of MinimaxQ (Littman '98)

Online Outcome Sampling (OOS)

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

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
  $\rightarrow$ 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
  $\rightarrow$ 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
  $\rightarrow$ MCTS version of outcome sampling MCCFR
- Use regret matching over sampled counterfactual regrets
- *Converges to Nash equilibrium over time!*

Win/Loss Goofspiel(13) Performance

P1 \ P2	RND	DUCT(max)	DUCT(mix)	Exp3	OOS	OOS^+
DUCT(max)	76.0					
DUCT(mix)	78.3	57.5				
Exp3	80.0	55.8	48.4			
OOS	73.1	55.3	43.8	47.0		
OOS^+	77.7	67.0	53.3	60.0	57.1	
RM	80.9	63.3	53.2	57.2	58.3	50.4

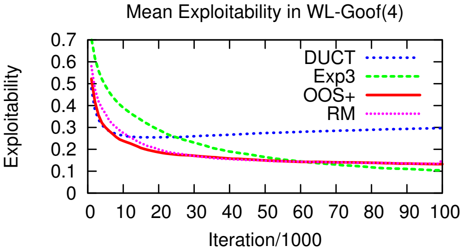
- Number refers to % win rate of row player type.

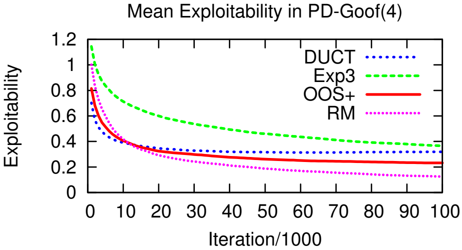
Point Difference Goofspiel(11) Exploitability

Algorithm	Mean Ex_2	Mean Ex_4	Mean simulations per second
DUCT(max)	7.43 ± 0.15	12.87 ± 0.13	124127 ± 286
DUCT(mix)	5.10 ± 0.05	7.96 ± 0.02	124227 ± 286
Exp3	5.77 ± 0.10	10.12 ± 0.08	125165 ± 61
OOS	$\mathbf{4.02} \pm \mathbf{0.06}$	$\mathbf{7.92} \pm \mathbf{0.04}$	186962 ± 361
OOS+	5.59 ± 0.09	9.30 ± 0.08	85940 ± 200
RM	5.56 ± 0.10	9.36 ± 0.07	138284 ± 249

- Lower Ex_d = closer to Nash eq.

WL-Goof(4) and PD-Goof(4) Full Exploitability

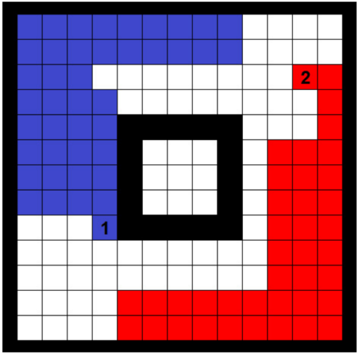




- x-axis is time, y-axis is distance to Nash eq. (lower = closer)

New Results I: Tron

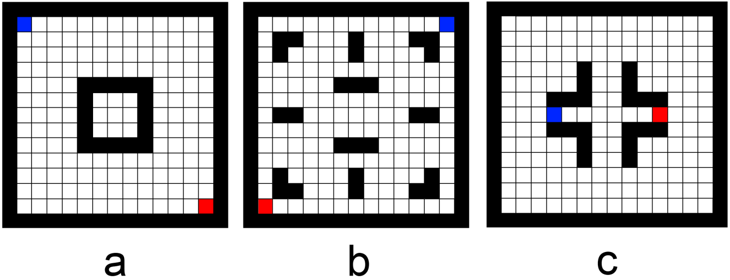
(Lanctot et al.'13). Based on Bachelor thesis of Christopher Wittlinger.



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

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

- Percentage is a win rate
- Results of the different variants played against each other
- $\pm$ refers to 95% confidence intervals.

New Results II: Consistency Guarantees

(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 ϵ-minmax in Goofspiel:

vs. DUCT(max)	75.70 %
vs. DUCT(mix)	48.60 %
vs. Exp3	78.50 %
vs. OOS	31.05 %
vs. OOS$^+$	55.75 %
vs. RM	47.55 %

- All are roughly ± 3.0 for 95% c.i.
- Must solve LP only so often
- Must decay ϵ
- Seems more sensitive to parameters

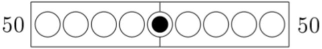
Preliminary Results II

In Tron, run RM with purification: if probability < 0.2, flush it to 0 and renormalize.

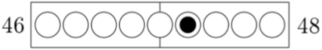
	RM	RM + purification
vs. DUCB1T(max)	31.52 %	35.07 %
vs. DUCT(max)	68.48 %	50.30 %

- RM(purification) wins 54.44% againsts RM
- All are roughly ± 2.0 for 95% c.i.

Preliminary Results III: Oshi-Zumo



[50,4,∗]–Oshi–Zumo starting position – code (50, 50, 0)



Position after move (4,2) – code (46, 48, 1)

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

	OZ [50,3,1]	OZ [15,3,1]
DUCT(mix)	16.60 %	36.30 %
Exp3	31.10 %	44.95 %
OOS	11.40 %	25.50 %
OOS$^+$	23.85 %	42.40 %
RM	41.80 %	58.60 %

- 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

Questions?

marc.lanctot@maastrichtuniversity.nl
mlanctot.info

Network and Strategic Optimisation (NSO) Group
project.dke.maastrichtuniversity.nl/nso/

This work is partially funded by the Netherlands Organisation for Scientific Research (NWO).

NWO
Netherlands Organisation for Scientific Research