Monte-Carlo Methods

Timo Nolle

TECHNISCHE UNIVERSITÄT DARMSTADT

Outline

- Minimax
- Expected Outcome
- Monte-Carlo
  - Monte-Carlo Simulation
  - Monte-Carlo Tree Search
  - UCT
  - AMAF
  - RAVE
  - MC-RAVE
  - UCT-RAVE
- Playing Games
  - Go, Bridge, Scrabble
- Problems with MC
  - Laziness
  - Basin structure
  - Dangers of Random playouts

Minimax

TECHNISCHE UNIVERSITÄT DARMSTADT

- Assume that black is first to move
- Outcome of a game is either
  - 0 = win for white
  - 1 = win for black
- Black wants to maximize the outcome
- White wants to minimize it
- Given a small enough game tree (or enough time) minimax can be used to compute perfect play
- Problem: this is not always possible

$$\mathtt{ev} : G \rightarrow \{\mathtt{max}, \mathtt{min}\} \cup [0,1]$$

$$\mathtt{ev}_c(p) = \begin{cases} \mathtt{ev}(p), & \textit{if } \mathtt{ev}(p) \in [0,1]; \\ \max_{p' \in s(p)} \mathtt{ev}_c(p'), & \textit{if } \mathtt{ev}(p) = \mathtt{max}; \\ \min_{p' \in s(p)} \mathtt{ev}_c(p'), & \textit{if } \mathtt{ev}(p) = \mathtt{min}. \end{cases}$$

$$\begin{aligned} &\textbf{if } \mathtt{ev}(p) \in [0,1] \textbf{ return } \mathtt{ev}(p) \\ &\textbf{if } \mathtt{ev}(p) = \mathtt{max} \textbf{ return } \max_{p' \in s(p)} \mathtt{minimax}(p') \\ &\textbf{if } \mathtt{ev}(p) = \mathtt{min} \textbf{ return } \min_{p' \in s(p)} \mathtt{minimax}(p') \end{aligned}$$

Expected Outcome

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bruce Abramson (1990)
  - Used the idea of simulating random playouts of a game position
  - Concerns: Minimax approach has several problems
    - Hard to calculate, hard to estimate
    - Misses precision to extend beyond two-player games
  - He proposes a domain-independent model of static evaluation
    - Namely the Expected Outcome of a game given random play

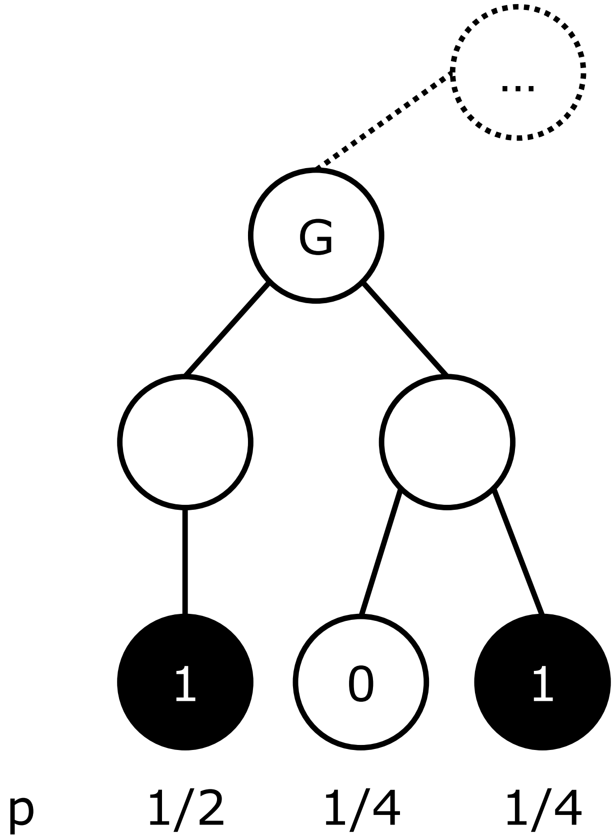
Expected Outcome

- Expected outcome is defined as

$$EO(G) = \sum_{leaf=1}^{k} V_{leaf} P_{leaf}$$

- where
  - G = game tree node
  - k = #leafs in subtree
  - V_{leaf} = leafs value
  - P_{leaf} = probability that leaf will be reached, given random play

Expected Outcome



$$EO(G) = \left(1 * \frac{1}{2} + 0 * \frac{1}{4} + 1 * \frac{1}{2}\right) = \frac{3}{4}$$

EO(G) is the chance of winning when playing the move that leads to G.

Expected Outcome

- Problem
  - The perfect play assumption is easily defended
  - The random play assumption on the other hand not really
    - A strong move against a random player does not have to be good against a real one
- Consider this hypothetical scenario
  - Two identical chess midgame positions
  - Human player has 1 minute to make a move on each board
  - Board 1 is then played out randomly, whereas board 2 is played out by (oracularly defined) minimax play
  - Obviously on board 1 the player wants to play the move with the highest EO, whereas on board 2 he wants to play the strongest minimax move

  - **Assumption: The correct move is frequently (not always) the same**

Expected Outcome

- Apply EO methods to games with incomplete information (e.g. games where the game tree is too big)
- Example Othello (8x8)
  - EO has to be estimated
  - Do this by sampling a finite number of random playouts
    - Sample N = 16 leaves, calculate $\mu_0 = \frac{WINS}{N}$
    - Then $\mu_1 = \frac{WINS}{2N}$, … $\mu_i = \frac{WINS}{2^i N}$ until $|\mu_i - \mu_{i-1}| \leq \epsilon$
- This Sampler beat Weighted-Squares (fairly strong) 48-2
  - Although it occasionally took >2h to make a move
  - Disk Difference shows that Sampler is a full class better than Weighted-Squares

Monte-Carlo Simulation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Monte-Carlo Tree Search and Rapid Action value Estimation in Computer Go (Gelly and Silver, 2011)

- Definition:

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s)} \mathbb{I}_i(s,a) z_i$$

- Where
  - N(s,a) = # of times action a was selected in state s, z_i = value of ith simulation, N(s) = the total number of simulations
  - $\mathbb{I}_i(s,a)$ = A function that return 1 if action a was selected in state s and 0 otherwise

Monte-Carlo Tree Search

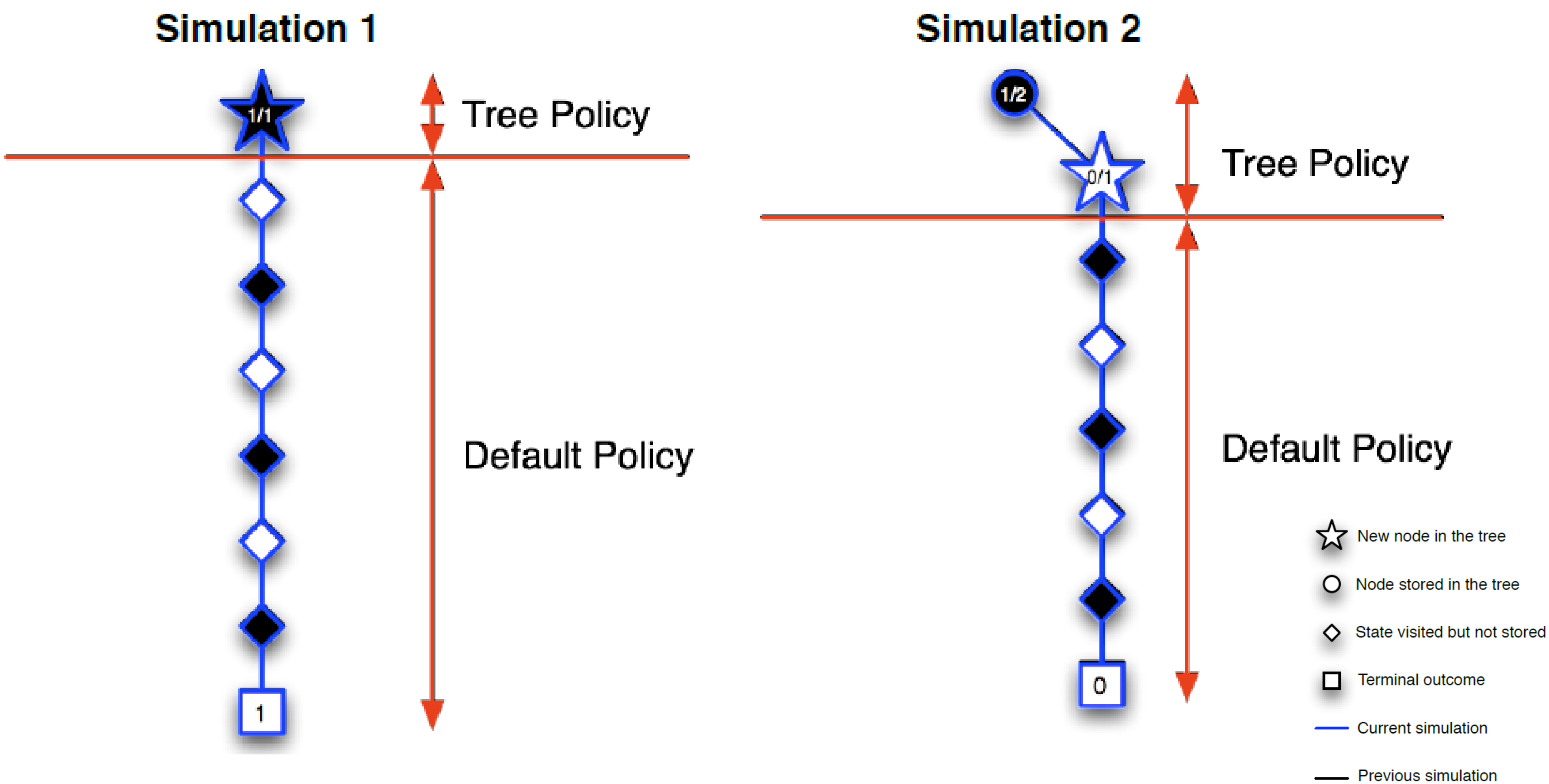
TECHNISCHE UNIVERSITÄT DARMSTADT

- Uses MC Simulation to evaluate the nodes of a search tree

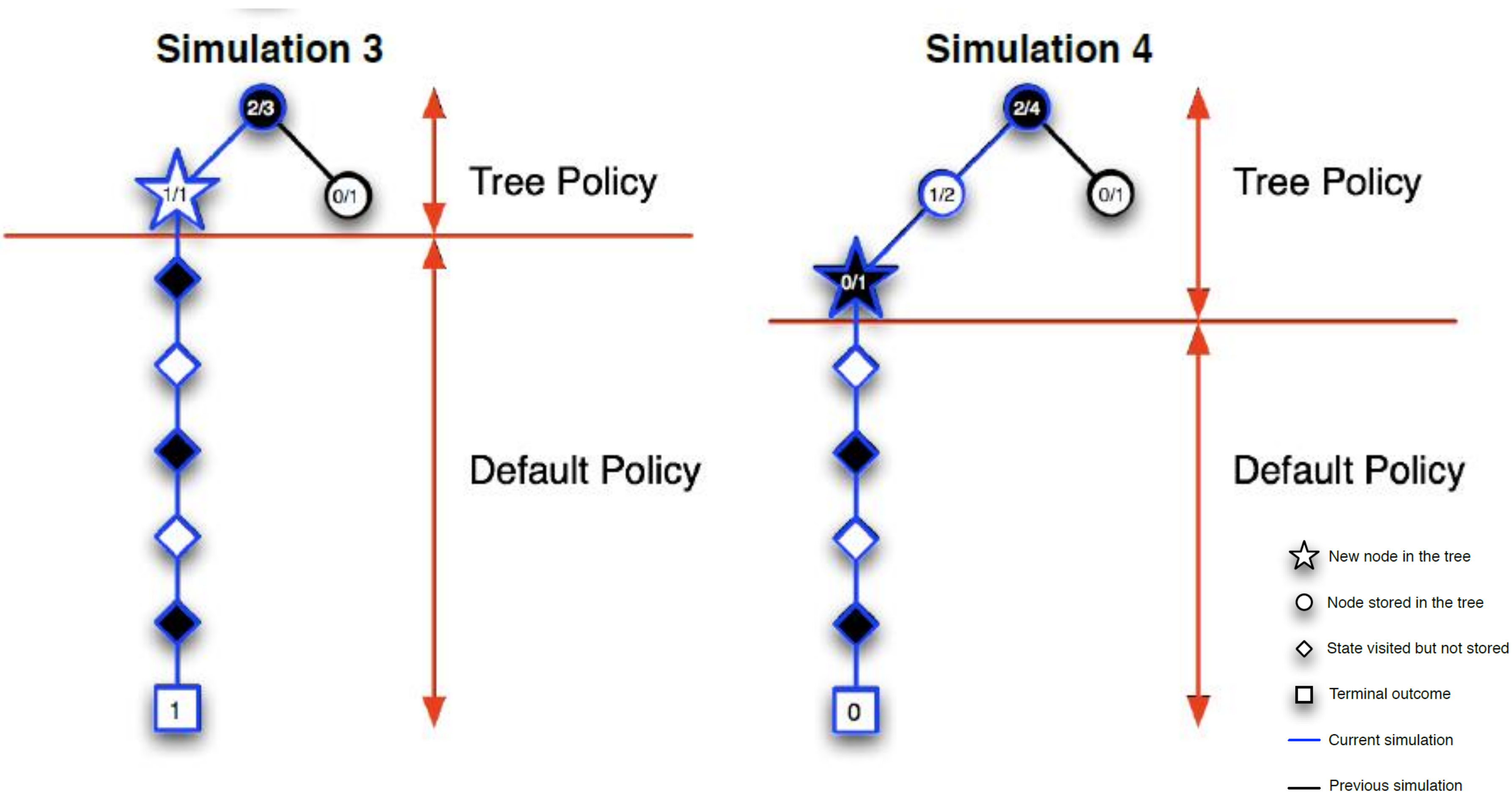
- How to generate
- Start with an empty Tree T
- Every simulation adds one node to the T (the first node visited that is not yet in the tree)
- After each simulation every node in T updates it's MC value

- As the tree grows bigger the node values approximate the minimax value (n → infinity)

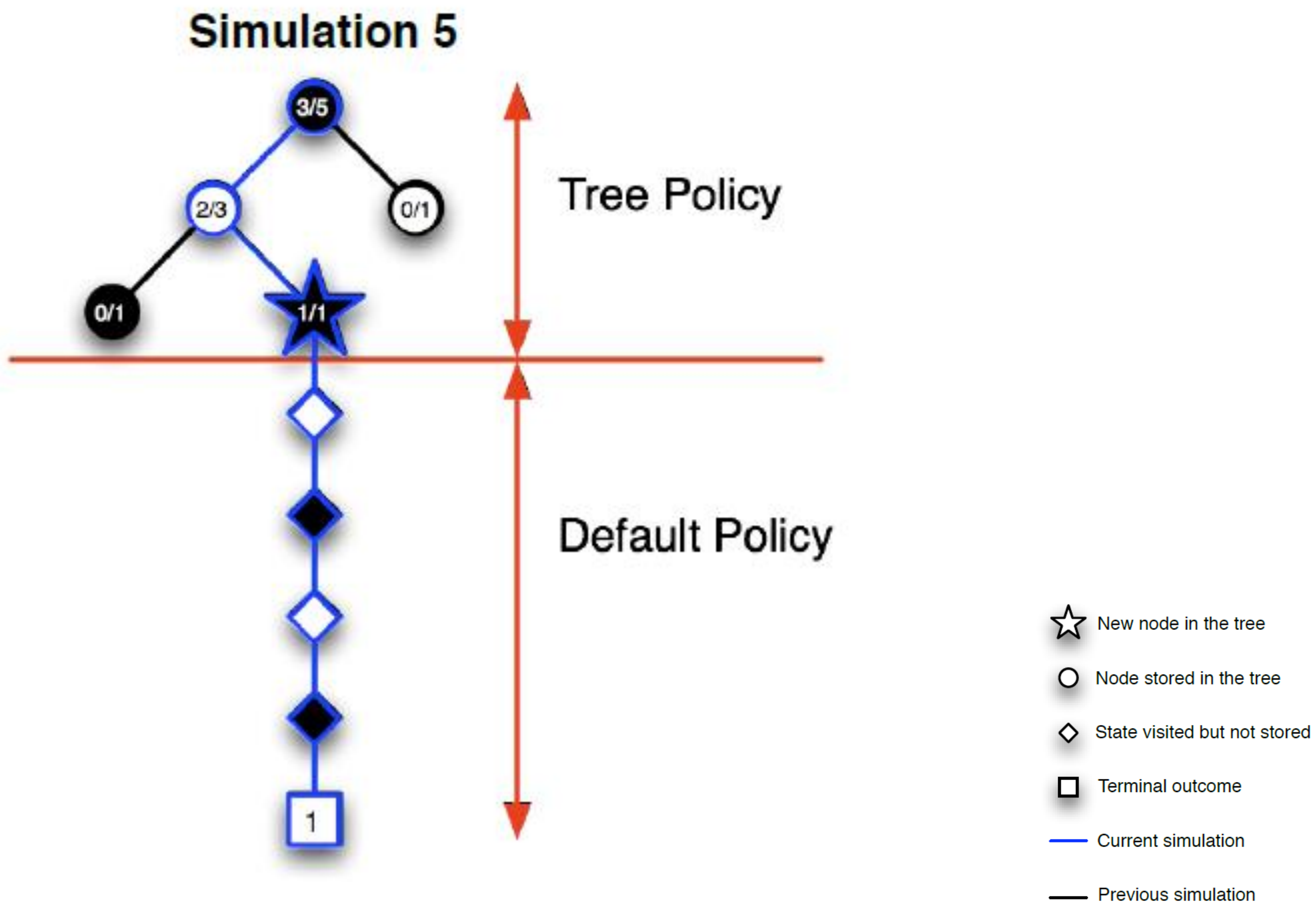
Monte-Carlo Tree Search



Monte-Carlo Tree Search



Monte-Carlo Tree Search



UCT (Upper Confidence Bound 1 applied to trees)

- Idea: optimism in case of uncertainty when searching the tree
  - Greedy action selection typically avoids searching actions after one or more poor outcomes

  - UCT treats each state of the search tree as a multi-armed bandit
    - The action value is augmented by an exploration bonus that is highest for rarely visited state-action pairs
    - The tree policy selects the action a* maximizing the augmented value

$$Q^{\oplus}(s,a) = Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}$$

$$a^* = \operatorname*{argmax}_a Q^{\oplus}(s,a)$$

c = Exploration constant

All Moves As First Heuristic

- MC alone can't generalize between related positions
- Idea: have one general value for each move independent from when it's played
- Combine all branches where an action a is played at any point after s

- MC simulation can be used to approximate the AMAF value

$$\tilde{Q}(s,a) = \frac{1}{\tilde{N}(s,a)} \sum_{i=1}^{N(s)} \tilde{\mathbb{I}}_i(s,a) z_i$$

- Gelly and Silver (2011) used this in computer Go

Rapid Action Value Estimation

- The RAVE algorithm uses the all-moves-as-first heuristic to share knowledge between nodes

  - As normal MC has to play out many games for any action in any state it is a good idea to save capacity by using the AMAF heuristic
    - Moves are often unaffected by moves played elsewhere on the board
    - → One general value for each move

  - Especially in GO the branching factor is very big

MC-RAVE

- The RAVE algorithm learns very quickly, but is often wrong
  - Idea: Combine the RAVE value with the MC value and make decisions based on that
  - Each node in the Tree then has an AMAF and a MC value

$$Q_\star(s,a) = (1-\beta(s,a))Q(s,a) + \beta(s,a)\tilde{Q}(s,a)$$

  - β is a weighting parameter for state s and action a
    - It depends on the number of simulations that have been seen
      - When only a few simulations have been seen the AMAF value has to be weighted more highly ($\beta(s,a) \approx 1$).
      - When many simulations have been seen, the MC value is weighted more highly($\beta(s,a) \approx 0$)
  - Heuristic MC-Rave: add a heuristic that initializes node values

UCT-RAVE

- Application of the optimism-in-the-face-of-uncertainty principle

$$Q_{\star}^{\oplus}(s,a) = Q_{\star}(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}$$

Playing Go

TECHNISCHE UNIVERSITÄT DARMSTADT

- For the last 30 years computers have evaluated Go positions by using handcrafted heuristics, based on human expert knowledge, patterns, and rules
- With MC no human knowledge about positions is required
- When heuristic MCTS was added to MoGo it was the first program to reach the dan (master) level and the first to beat a professional player
- Traditional programs rated about 1800 Elo
- MC programs with RAVE rated about 2500 Elo
- After initial jump of MC programs in go
  - Computer programs improved about one rank every year

Playing Bridge

- GIB: Imperfect Information in a Computationally Challenging Game (Ginsberg, 2001)
- Used MC for generating deals that are consistent with both the bidding and the play of the deal thus far
- MC used for card play and bidding (though reliant on big bidding database)

Playing Scrabble

- World-championship-caliber Scrabble (Sheppard, 2001)
- MAVEN on of the first programs to employ simulated games for the purpose of positional analysis
- Different algorithms for early-, mid-, and endgame
  - Problem the earlygame engine favors move A, but the midgame engine prefers another move B
  - Simulation can be used here to get an answer
  - Just simulate out the game after the moves A and B and calculate the winning probability for each move

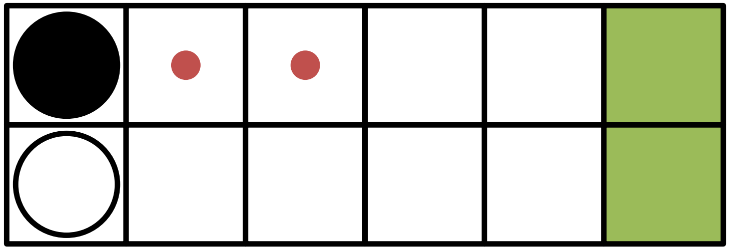
Playing Scrabble

- MAVEN averages 35.0 points per move, games ar over in 10.5 moves and MAVEN plays 1.9 bingos per game
- Human experts average 33.0 points, 11.5 moves per game and about 1.5 bingos per game
- MAVEN is stronger than any human

Problems with MC

- On the Laziness of Monte-Carlo Game Tree Search in Non-tight situations (Althofer, 2008)

- Invented the double step race
  - Every move you are allowed to move either 1 or 2 squares
  - The player first to reach the green square wins
  - For a human the optimal strategy is obvious: always move 2 squares except you are 1 square away from the finish
  - The figure shows an example of a 6-vs-6 Double Step Race



Problems with MC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Experiment
  - Look at different DSRs (6-vs-3 – 6-vs-10), play 10.000 games.
  - The game tree was generated by playing n random games from the starting position
  - Table: Number of wins for Black (from 10.000 games)

n	6-vs-3	6-vs-4	6-vs-5	6-vs-6	6-vs-7	6-vs-8	6-vs-9	6-vs-10
1	361	2204	4805	6933	9340	9649	9967	9988
2	193	1987	5799	6970	9654	9752	9991	9996
4	47	1468	7039	7450	**9905**	**9868**	9999	9999
8	2	836	**8573**	**8228**	**9989**	**9969**	**10000**	**9999**
16	0	286	**9557**	**9264**	**10000**	**9992**	10000	10000
32	0	40	**9936**	**9875**	10000	10000	10000	10000
64	0	0	**10000**	**9991**	10000	10000	10000	10000

Problems with MC

- Table: Number of games with a bad single step on the first move for Black

n	6-vs-3	6-vs-4	6-vs-5	6-vs-6	6-vs-7	6-vs-8	6-vs-9	6-vs-10
1	4714	3992	**3511**	**3732**	4259	4798	4935	4954
2	4363	3270	**2797**	**2993**	3625	4493	4803	4955
4	3864	2368	**2091**	**2081**	2870	3988	4755	4897
8	2968	1319	**1131**	**1281**	1835	3331	4502	4920
16	1775	568	**396**	**465**	0831	2399	4185	4912
32	642	118	**64**	**86**	250	1334	3395	4687
64	80	9	**0**	**6**	32	438	2339	4453

- Evaluation: MC performs better in tight positions, when Black already has an advantage, it tends to be “lazy”

Problems with MC

- Game Self-Play with Pure Monte-Carlo: The Basin Structure (Althofer, 2010)
  - Continuation on the first paper
  - Self play experiments
    - One two MC players (with different MC parameters) play the Double Step Race
      - MC(k) vs MC(2k) for example

DSR-6

MC(k) vs MC(2k)	Number of games	Score
1-2	999.999	42.4 %
2-4	**999.999**	**40.9 %**
3-6	999.999	41.0 %
4-8	999.999	41.4 %
5-10	999.999	41.9 %
6-12	999.999	42.3 %
8-16	999.999	43.5 %
16-32	100.000	46.6 %
32-64	100.000	49.4 %
64-128	100.000	50.0 %

DSR-10

TECHNISCHE UNIVERSITÄT DARMSTADT

MC(k) vs MC(2k)	Number of games	Score
1-2	999.999	41.6 %
2-4	999.999	40.1 %
4-8	999.999	39.3 %
8-16	**999.999**	**38.8 %**
16-32	999.999	41.6 %
32-64	999.999	46.9 %
64-128	999.999	49.6 %
128-256	999.999	50.0 %

DSR-14

MC(k) vs MC(2k)	Number of games	Score
1-2	100.000	40.4 %
2-4	100.000	39.0 %
4-8	100.000	37.9 %
8-16	100.000	**36.9 %**
16-32	100.000	37.3 %
32-64	100.000	43.0 %
64-128	100.000	48.3 %
128-256	100.000	49.9 %

DSR-32

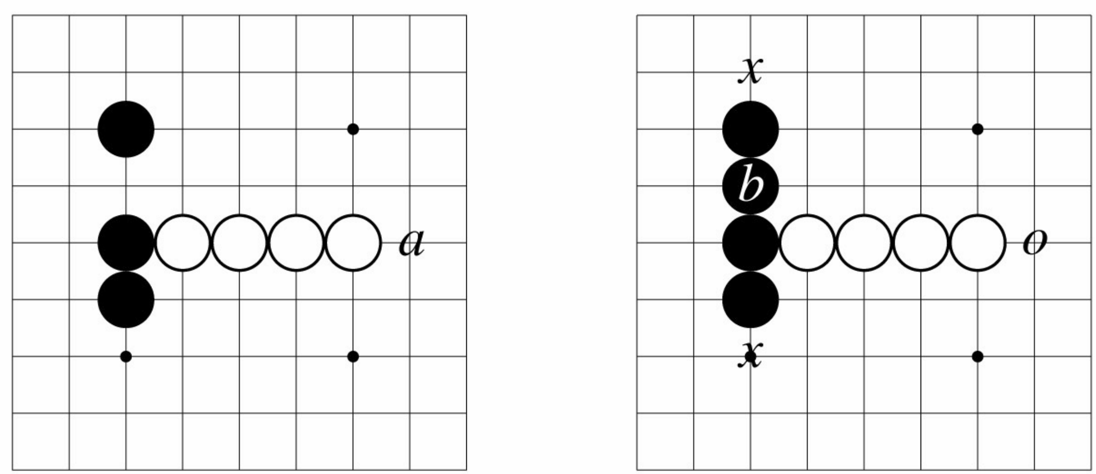
MC(k) vs MC(2k)	Number of games	Score
1-2	10.000	38.7
2-4	10.000	35.6
4-8	10.000	33.1
8-16	999.999	30.9
16-32	**100.000**	**30.1**
32-64	110.000	30.3
64-128	10.000	34.9
128-256	10.000	44.8

Evaluation

- This “basin” structure also appeared in other games
  - E.g., Clobber, conHex, “Fox versus Hounds”, “EinStein wurfelt nicht”
  - Some even had double basins
- “… it is not clear which applications the knowledge about the existence and shape of self-play basins will have.”

Problems with MC

- More simulations do not always lead to better results (Browne, 2010)
  - For example in a game of Gomoku flat MC fails to find the right move a
    - Problem is move b creates two next-move wins for Black as opposed to one next-move win for white even though it is White’s turn next move
    - This improves however when using tree search
    - Though it takes a long time to converge



QUESTIONS

References

- Expected-Outcome: A General Model of Static Evaluation, Abramson, 1990
- On the Laziness of Monte-Carlo Game Tree Search in Non-tight Situations, Althöfer, 2008
- Game Self-Play with Pure Monte-Carlo: The Basin Structure, Althöfer, 2010
- On the Dangers of Random Playouts, Browne 2010
- Monte-Carlo tree search and rapid action value estimation in computer Go, Gelly and Silver, 2011
- GIB: Imperfect Information in a Computationally Challenging Game, Ginsberg, 2001
- World-championship-caliber Scrabble, Sheppard, 2002