Monte-Carlo Tree Search Applied to the Game of Havannah

J. D. Fossel

September 12, 2010

Abstract

Playing the game of Havannah is difficult for computers. Nevertheless, Monte-Carlo Tree Search (MCTS) has been successfully applied to play Havannah in the past. In this paper several enhancements to the selection and play-out phases of MCTS are applied to increase the playing strength.

For the selection phase, we investigate the following three enhancements: (1) Rapid Action-Value Estimate (RAVE), (2) Progressive History, and (3) Extended RAVE, a combination of RAVE and Progressive History.

For the play-out phase, we investigate four play-out strategies: Favoring (1) playing on corner cells, (2) moves that are in proximity to previous moves, (3) moves that lead to a forced win, and (4) moves that have positive RAVE-values.

The experiments reveal that RAVE, Progressive History, and Extended RAVE increase the win rate, whereas Progressive History gives the best results. Checking for forced wins also improves the play-outs, while the other tested play-out strategies do not.

1 Introduction

Humans have played board games for thousands of years. With the upcoming of computers, humans wrote programs for playing games. The traditional approach is $\alpha\beta$ search [8] with a positional evaluation function, which searches the state-space of a game in form of a search tree. Though, if the game-tree is too complex and no suitable evaluation function is known, this approach is not applicable. In that case Monte-Carlo Tree Search (MCTS) might be a viable option because MCTS requires no evaluation function to estimate the game-theorethic value of a position.

In this paper, we investigate Havannah, a game with high branching factor for which no strong evaluation function is known. We develop a MCTS player for Havannah and investigate techniques to increase its playing strength.

The research questions of this paper are:

- Does Rapid Action-Value Estimate (RAVE), Progressive History, or Extended RAVE increase the playing strength of MCTS in Havannah?
- Which play-out strategies lead to an increase in playing strength?

This paper is structured as follows: In Section 2 the rules of Havannah are explained. MCTS is presented in Section 3. Afterwards, enhanced selection strategies are introduced in Section 4, and play-out strategies in Section 5. Subsequently, experiments and results are given in Section 6. Finally, in Section 7 general conclusions are drawn and future research is suggested.

2 The game of Havannah

Havannah [6] was invented by the Dutch game designer Christian Freeling in 1979. It is a two-player, zero-sum, discrete, finite and deterministic game of perfect information played on a hexagonal board. While in tournaments this board is of base-10 (i.e. each side has length 10), smaller board sizes are available for beginners. Both players successively place a stone of their color (White or Black) on the board until there is no free cell left, resulting in a draw, or until a chain of stones forms a winning connection. There are three winning connections (see Figure 2.1):

- A *fork* is a chain of stones linking 3 sides (corners do not count).
- A *ring* is a chain of stones around at least one cell (empty or not does not matter).
- A *bridge* is a chain of stones linking 2 corners.

Freeling offers a prize of 1,000 euros for any computer program that is able to beat him in one out of ten matches on a base-10 board before 2012. But while humans are strong Havannah players, it is a difficult game for computers because of the absence of the following elements:

- No material imbalance [9]
- No general direction [9]
- Only few local patterns are known [16]

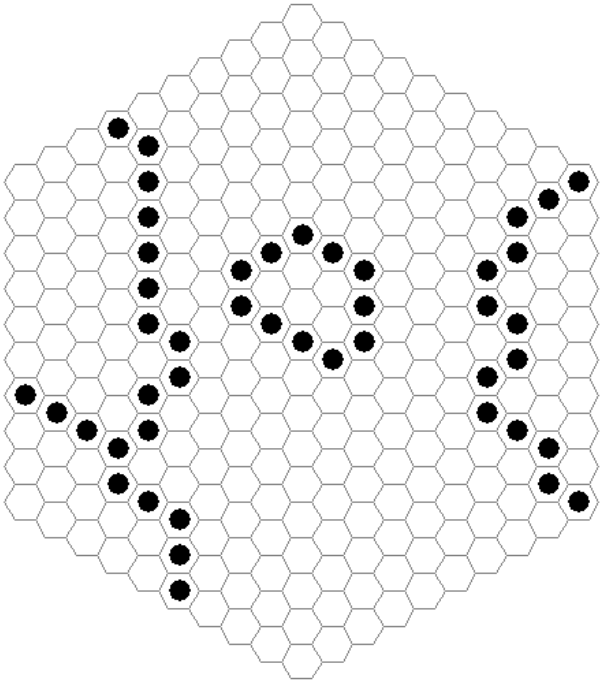

Figure 2.1: Base-10 board with the three winning connections, fork, ring, and bridge

- No strong evaluation function exists [16]
- No safe pruning heuristic to reduce the number of moves exists [16]
- High game-tree and state-space complexity of $10^{157}$and 10^{127} [9], respectively. (Chess for comparison has a game-tree complexity of 10^{123} and a state-space complexity of 10^{46} [18].)

Therefore traditional search algorithms are expected to perform poorly. Recently, Monte-Carlo Tree Search (MCTS) [5, 10] lead to a decent playing strength on smaller board sizes, e.g. the WANDERER [11] programm developed by Lorentz.

3 Monte-Carlo Tree Search

For games where no strong evaluation function exists, Monte-Carlo Tree Search (MCTS) [5, 10] may be an adequate alternative. MCTS constructs a search tree by using simulated games for evaluating a position. It has been successfully applied to many games, examples are computer Go [5] and Lines of Action [19].

In MCTS, nodes represent the position (board state) of the game, where each node i contains the value v_i, the number of times it has been visited n_i and the score $s_i = v_i/n_i$. v_i is increased by 1 for every play-out won, and decreased by 1 for every play-out lost.

MCTS consists of four phases shown in Figure 3.1: Selection, play-out, expansion and back-propagation.

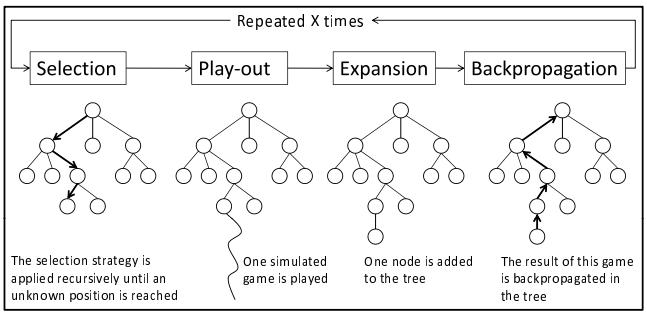

Figure 3.1: The four MCTS phases [3]

Selection: The first phase consists of selecting a node for play-out. Two possibilities arise: (1) either select a node that while currently being not the best might turn out to be good (exploration), or (2) select a well performing node and confirm/refute that the associated move is good (exploitation).
Choosing a good balance between exploration and exploitation is crucial. Because if on the one hand, the selection is too explorative the tree will become too shallow to allow tactical play. If, on the other hand, it is too exploitative the algorithm might get stuck in a local maximum.
The strategy UCT (Upper Confidence Bound applied to Trees) [10] can be used to balance exploration and exploitation. It has its origin in the Multi-Armed Bandit problem [13], a well studied problem that deals with similar exploration/exploitation balancing issues. UCT uses Formula 3.1 to select which child k of node p should be selected, whereas I is the set of nodes that are children of node p.

$$k \in argmax_{i \in I}(s_i + C \cdot \sqrt{\frac{ln\, n_p}{n_i}}) \qquad (3.1)$$

C is a constant that determines the balance between exploration and exploitation (the higher C is set, the more explorative the selection will be). The node that satisfies Formula 3.1 is selected for expansion. In practice however, the selection strategy is only applied in nodes, which visit count exceeds a certain threshold T [5] ($T = 50$ for Havannah [9]). If this is not the case, the play-out strategy is used for selecting a node.

Play-out: When a node has been selected, and the selected node does not represent a terminal position, a play-out is performed. This play-out is done according to a certain strategy. This strategy either

chooses random moves that have not been played yet, or uses domain knowledge to increase the quality of the play-outs.

Expansion: After the play-out, a node is added as child of the selected node. The move and value of that node are set according to the result of the play-out. Furthermore, to prevent the algorithm from getting stuck in a local maximum, all siblings that represent a move not played yet are also added. The value of these nodes is set to 1, so that the algorithm will consider them worth visiting in the future.

Backpropagation: After expanding, the value of the expanded node is backpropagated to the root node, whereas the visit count of every node that is traversed is increased by one.

These four phases are repeated either a certain number of times, or for a certain duration. Afterwards the most promising move (e.g. the one with the highest visit count, most wins, or highest score) is executed. In this implementation the move regarded best is the one with the highest score s.

4 Alternative Selection Strategies

As the value v gets more reliable with increasing number of visits, it may be beneficial to use additional information in the selection phase if nodes have a small visit count. Doing so has turned out to increase the playing strength in multiple domains, e.g. in Go [15] and Chinese Checkers [12].

In this section three selection strategies for MCTS are presented. Subsection 4.1 introduces Rapid Action-Value Estimate (RAVE), Subsection 6.1.3 Progressive History (PH), and Subsection 4.3 Extended RAVE (ER).

4.1 Rapid Action-Value Estimate

Rapid Action-Value Estimate (RAVE) [7] is a way to combine the All-Moves-as-First (AMAF) [1] heuristic with UCT.

The basic idea of AMAF is incorporating additional information when determining which node to select. Hence, in addition to s_i (Formula 3.1), also r_i, the average value of all nodes with the same move (as i) that are descendants of node p (the parent of node i), are taken into account when selecting a node for play-out.

RAVE uses AMAF to acquire results from the play-outs quicker. Its impact is reduced with increasing number of visits because it provides less reliable scores than UCT.

RAVE is applied to Havannah [16] by using Formula 4.1 to determine which child k of node p is selected for expansion, whereas I is the set of nodes that are children of node p.

$$k \in argmax_{i \in I}(\beta'_i \cdot s_i + \beta_i \cdot r_i + C \cdot \sqrt{\frac{ln\, n_p}{n_i}}) \quad (4.1)$$

β_i and β'_i are given in Formula 4.2 and Formula 4.3 [16].

$$\beta_i = R/(R + n_i) \quad (4.2)$$

$$\beta'_i = 1 - \beta_i \quad (4.3)$$

The higher the constant R, the longer AMAF will be considered.

4.2 Progressive History

Like RAVE, Progressive History [12] addresses the problem that it may take some time to gather enough information to determine a reliable value for a node. Progressive History is a combination of Progressive Bias [2] and the history heuristic [14, 20]. Progressive Bias requires heuristic knowledge. The Progressive Bias part in Progressive History is using $W/(n_i - v_i + 1)$ to bias the history heuristic part pv_i/pn_i (see Formula 4.4 [12]). pv_i is the summed value of move i for every time i has been played, and pn_i is the total number of times move i has been played.

The child k of node p that satisfies Formula 4.4 is selected for expansion (I is the set of the children of p), whereas ph_i (Formula 4.5) is the Progressive History part:

$$k \in argmax_{i \in I}(s_i + C \cdot \sqrt{\frac{ln\, n_p}{n_i}} + ph_i) \quad (4.4)$$

$$ph_i = \frac{pv_i}{pn_i} \cdot \frac{W}{n_i - v_i + 1} \quad (4.5)$$

The parameter W controls the Progressive Bias, i.e. how the influence of the Progressive History behaves. By dividing W through $(n_i - v_i + 1)$ when determining the Progressive Bias, nodes with low score are not biased for long, and nodes with high score continue to be biased.

4.3 Extended RAVE

We propose Extended RAVE, a combination of RAVE and Progressive History. The difference to Progressive History is that we use er_i (see Formula 4.7) for the history heuristic part instead of pv_i/pn_i. er_i consists of two parts: (1) r_k, the usual RAVE score in the parent p of node k, and (2) r'_k, the RAVE score in the parent of p (for the same move as k). r'_k is used because of the assumption that moves that where good in a similar position could also be good in the current one. To select a

child, node k that satisfies Formula 4.6 is choosen (I is the set of $p's$ children):

$$k \in argmax_{i \in I}(s_i + C \cdot \sqrt{\frac{ln\, n_p}{n_i}} + er_i) \tag{4.6}$$

$$er_i = (\beta_i \cdot r'_i + \beta'_i \cdot r_i) \cdot \frac{W}{n_i - v_i + 1} \tag{4.7}$$

whereas β_i and β'_i are given in Formula 4.2 and 4.3. The constant R defines the balance in influence between the standard RAVE (r_i) and extended RAVE (r'_i) component. Similar to Progressive History (Subsection 6.1.3), W controls the Progressive Bias in Extended RAVE that determines how much influence the enhancement has.

5 Play-out Strategies

Improving the play-outs may have two benefits: (1) More reliable values v may be achieved, and (2) the number of turns required to finish a play-out may be decreased, reducing the time required to simulate a game. However, sophisticated play-out strategies have the potential to be CPU intensive, and therefore may also decrease the number of play-outs per second.

This section presents the following enhancements to the MCTS play-out: Havannah-Mate is introduced in Subsection 5.1, RAVE Play-out in Subsection 5.2, Biased Play-out Corner in Subsection 5.3 and Biased Play-out Proximity in Subsection 5.4.

5.1 Havannah-Mate

The Havannah-Mate strategy proposed by Lorentz [11] checks if the play-out can be ended within the next two turns (each placing of a stone counts as a turn). If this check, which resembles a local 3-ply search, is successful it selects the according move to end the game next round (two turns). Unfortunately, determining whether a move leads to a forced win requires a rather large number of calculations because many cells need to be checked:

The Havannah-Mate enhancement needs to check all free cells adjacent to the subgraph the latest move is part of, and also the cells adjacent to these free cells. As the game progresses, the number of cells that need to be checked increases, as the subgraphs have a high probability of getting longer the more stones are placed on the board.

Figure 5.1 shows an example. Assuming that (A) is the most recent move, all free cells adjacent to the group of black stones it connects with, are marked black. After that, all cells that are adjacent to the cells marked black, are marked grey. If there are two cells marked black adjacent to a cell marked grey and occupied by the Black player (B), a connection between (B) and the group of (A) can always be made. If a victory condition is fullfilled using this virtual connection, the Black player will be able to terminate the game in three turns.

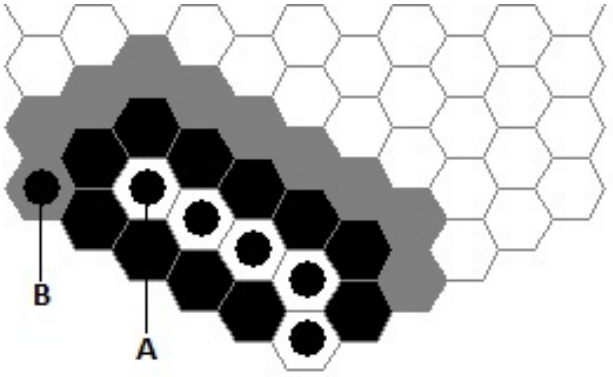

Figure 5.1: Necessary Havannah-Mate checks

5.2 RAVE Play-out

We propose RAVE Play-out, a strategy that uses RAVE values to bias the play-out. With a certain probability $P_{ravePlay}$, RAVE Play-out is triggered. In that case instead of randomly choosing a move, a move i with a positive RAVE value r_i in node p is selected instead, whereas node p is the parent of the extended node k. If no such node exists, a random move is played. If there are multiple moves with positive RAVE values, a move is chosen randomly whereas the probability for each move is determined by its RAVE value. If there are for example three RAVE values available, $r_0 = 2$, $r_1 = 3$ and $r_2 = -1$, with probability $P_{ravePlay}$ instead of randomly choosing a move with equal probability for each move, move 0 will be selected with probability $2/5$, and move 1 with probability $3/5$.

5.3 Biased Play-out Corner

We propose Biased Play-out Corner to utilize the fact that playing the corner cells is stronger than playing other cells due to the bridge winning condition. Therefore, the likelihood of playing corner cells is increased by weight w_c.

5.4 Biased Play-out Proximity

In Havannah, often stones are placed in proximity to other recently placed stones. Therefore we propose the Biased Play-out Proximity strategy, that assumes that playing on cells close to previous moves is strong. Thus the probability of selecting these cells in the play-out is increased by weight w_p.

A node is considered in proximity if it is not more than P_{range} cells away to at least one of the last H moves. $H = 1$ implies considering only moves close to the last move of the opponent for increasing the chance of being selected, while $H = 2$ also includes the last own move.

The additionally required operations for this modification are checking $6 \cdot H$ cells and using a random

generator to decide whether and on which of these cells should be played.

As an alternative mode, instead of increasing the weight of cells in proximity for every move in the play-out, Biased Play-out Proximity is used in a similar way to RAVE Play-out: With a certain probability P_{prox}, instead of considering all possible moves, only moves that satisfy the above stated proximity requirements are considered possible moves. If there are no moves available satisfying the proximity requirements, a random move is played.

6 Experiments

In this section experiments comparing the enhancements proposed in Section 4 and Section 5 for MCTS are conducted. The experiments are performed on a 2.4GHZ AMD Opteron CPU with 8GB of RAM. Because the starting player has an advantage, the colors assigned to the players alternate every game. The settings for the experiments, if not mentioned otherwise, are playing $n = 500$ games on a board of size $s = 6$ with both players allowed to take $t = 5$ seconds per move.

Experiments with selection strategies are reported in Subsection 6.1, experiments with play-out strategies in Subsection 6.2. In Subsection 6.3 a MCTS player with Progressive History selection and Havannah-Mate play-out is tested.

6.1 Selection Strategy Experiments

In this subsection different selection strategies are tested using random play-out. In Subsection 6.1.1 the C-constant for a MCTS-UCT player is tuned. In Subsection 6.1.2 RAVE is tested, Progressive History in Subsection 6.1.3, and Extended RAVE in Subsection 6.1.4.

6.1.1 Tuning UCT

In this subsection, experiments are conducted to determine the optimal C constant for UCT . According to Joosten [9], $C = 1.2$ is a good value when playing on board of size $s = 5$ with $t = 5$ seconds of time for each move. Yet, the optimal C value changes with different settings. Furthermore, to ensure reproducibility the C constant is not tuned for a fixed amount of time, but for a fixed number of play-outs.

Table 1 shows the win rate for different C values against a MCTS-UCT player with $C = 1.2$, whereas both players simulate 20,000 play-outs per turn. The best value for C is 1.8, winning 83% of the games against a UCT player with $C = 1.2$. It takes the MCTS-UCT player about 4.8 seconds to run 20,000 play-outs per turn.

C	1	1.4	1.6	1.8	2	2.5	3
Win	0.48	0.57	0.76	*0.83*	0.60	0.44	0.27

Table 1: MCTS-UCT parameter tuning, 20,000 play-outs per move

6.1.2 RAVE

To test RAVE, a MCTS-RAVE player is matched against a regular MCTS-UCT player with UCT constant $C = 1.8$. Instead of using a fixed number of play-outs, both players play-out phase last for $t = 5$ seconds per turn. The reason for this is that enhancements like RAVE may in- or decrease the cpu time required to simulate a game. Because Teytaud and Teytaud [16] successfully used $R = 50$ in their research for a slightly different setup, $R = 50$ is also used in this research.

Table 2 shows the win rate for MCTS-RAVE with different C values playing against MCTS-UCT. The result is that $C = 0.4$ performs best among the tested values for C, achieving a win rate of 67%. When playing White, the win rate is always at least 6% higher than when playing black. This confirms that the starting player (i.e. the White player) indeed has an advantage as mentioned before.

C	Win White	Win Black	Win total
0	0.53	0.47	0.62
0.1	0.53	0.47	0.64
0.2	0.55	0.45	0.63
0.3	0.58	0.42	0.64
0.4	0.54	0.46	*0.67*
0.5	0.59	0.41	0.62
0.75	0.57	0.43	0.60
1	0.59	0.41	0.57
1.25	0.56	0.44	0.52
1.5	0.56	0.44	0.48
2	0.54	0.46	0.42
4	0.60	0.40	0.37

Table 2: Win rate MCTS-RAVE vs. MCTS-UCT, $R = 50$

We also test MCTS-RAVE with $R = 125$ for $n = 250$ games. Table 3 shows that it does not increase the playing strength for these parameter settings.

In terms of runtime, MCTS-RAVE is slightly slower than MCTS-UCT, being able to simulate on average around 19,000 games in 5 seconds, whereas MCTS-UCT manages to simulate around 21,000 games.

6.1.3 Progressive History

To test Progressive History, MCTS-Progressive History (MCTS-PH) with various W and C is matched against MCTS-RAVE with $C = 0.4$ and $R = 50$. The result for this experiment can be seen in Table 4. For $W = 10$, $C = 0.2$ and $C = 0.3$ the win rate is 61% against MCTS-

C	Win White	Win Black	Win total
0	0.52	0.48	0.28
0.1	0.53	0.47	0.40
0.2	0.51	0.49	0.44
0.3	0.56	0.44	0.43
0.4	0.59	0.41	0.45
0.75	0.63	0.37	0.45
1	0.53	0.47	*0.46*
1.25	0.58	0.42	0.44
1.5	0.63	0.38	*0.46*
1.75	0.60	0.40	0.39

Table 3: Win rate MCTS-RAVE vs. UCT, $R = 125$

RAVE. When matched against MCTS-UCT ($C = 1.8$), MCTS-PH achieves a win rate of 73%.

Like MCTS-RAVE, MCTS-PH is able to run approximately 19,000 play-outs in 5 seconds.

C	$W = 1$	$W = 3$	$W = 5$	$W = 10$	$W = 15$
0	0.51	0.49	0.52	0.54	-
0.1	0.53	0.54	0.54	0.56	-
0.2	0.54	0.56	0.58	*0.61*	0.56
0.3	*0.58*	*0.60*	0.57	*0.61*	*0.61*
0.4	0.57	0.58	*0.59*	0.60	0.57
0.5	0.55	0.58	0.57	0.58	0.58
0.6	0.56	0.59	0.56	0.54	-
0.7	0.50	0.53	0.55	0.56	-
1	0.49	0.45	0.49	0.51	-

Table 4: Win rate MCTS-PH vs. MCTS-RAVE

6.1.4 Extended RAVE

To test Extended RAVE, MCTS-Extended RAVE (MCTS-ER) is matched against MCTS-RAVE. The parameters for MCTS-RAVE are set as following: $C = 0.4$ and $R = 50$. MCTS-Extended RAVE also uses $R = 50$, but is tested for various C values. As can be seen in Table 5, MCTS-ER outperforms MCTS-RAVE for $W = 10$ and $C = 0.4$ with a win rate of 60%.

MCTS-Extended RAVE is also hardly slower than RAVE. Like MCTS-RAVE, MCTS-ER is able to run approximately 19,000 play-outs in 5 seconds.

6.2 Play-out Strategy Experiments

In this subsection different play-out Strategies are tested. The default setup is matching MCTS-RAVE with the particular play-out strategy that is being tested against MCTS-RAVE with random play-out. The parameters for RAVE are $C = 0.4$ and $R = 50$.

Havannah-Mate is tested in Subsection 6.2.1, RAVE Play-out in Subsection 6.2.2, Biased Play-out Corner in Subsection 6.2.3 and Biased Play-out in Subsection 6.2.4.

C	$W = 1$	$W = 3$	$W = 5$	$W = 10$	$W = 15$
0	0.49	-	-	-	-
0.1	0.52	0.52	-	-	-
0.2	0.51	0.49	0.54	0.55	-
0.3	*0.53*	0.51	0.53	0.58	0.53
0.4	0.52	0.52	*0.56*	*0.60*	*0.60*
0.5	0.52	*0.53*	0.55	0.55	0.52
0.75	0.50	0.50	0.46	0.48	-
1	0.44	-	-	-	-
1.5	0.40	-	-	-	-

Table 5: Win rate MCTS-ER vs. MCTS-RAVE

6.2.1 Havannah-Mate

To evaluate the Havannah-Mate (HM) strategy, both players simulate 10,000 play-outs per turn.

As one can see in Table 6, using the Havannah-Mate strategy gives a win rate of 73%. But the strategy also requires about four times as much time as RAVE with random play-out needs to simulate the same number of play-outs.

	random	HM
win rate	0.28	0.72
time/turn	2.8 seconds	11 seconds

Table 6: Win rate MCTS-RAVE with Havannah-Mate play-out (HM) vs. MCTS-RAVE with random play-out, 10,000 play-outs per turn

Figure 6.1 shows the average play-out length of MCTS-RAVE with Havannah-Mate play-out and MCTS-RAVE with random play-out. Using Havannah-Mate consistenly leads to a lower or equal play-out length, presumably due to choosing better moves when simulating a game.

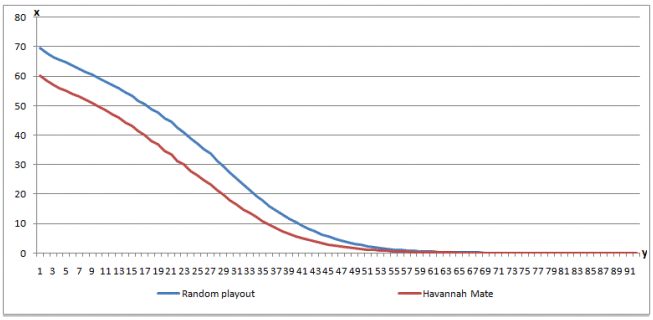

Figure 6.1: Play-out length (y-axis) MCTS-RAVE with Havannah-Mate play-out vs. MCTS-RAVE with random play-out depending on progression of the game (x-axis)

The difference in play-out length between random

play-out and Havannah-Mate play-out can be seen in Figure 6.2. The graph shows that the play-out advantage of Havanna-Mate becomes smaller as the game progresses. Therefore it seems reasonable to apply the Havannah-Mate strategy only for the first $MaxTurns/c$ turns when using time constraints, replacing it with random play-out afterwards. $MaxTurns$ is the maximum number of moves possible, i.e. the number of cells on the board.

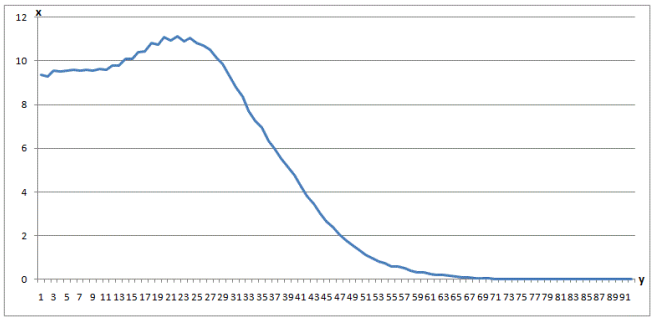

Figure 6.2: Play-out length advantage using Havannah-Mate (y-axis) depending on progression of the game (x-axis)

The results for $t = 5$ seconds and limiting the Havannah-Mate strategy to the first $MaxTurns/c$ turns can be seen in Table 7. When applying Havannah-Mate throughout the whole play-out ($c = 1$) it produces unsatisfactory results. For $c = 3$, i.e. if Havannah-Mate is only used for the first third of turns a win rate of 59% can be achieved. In that case the slowdown is only 21%. The reason for the drastic speed improvement is that checking for a Havannah-Mate becomes non linearly harder the more stones are placed on the board because potential winning connections that need to be checked become longer as the game progresses.

c	#play-outs random	#play-out HM	Win HM
1	20,130	5,086	0.33
1.5	19,918	6,454	0.35
3	21,671	17,169	*0.59*
4.5	21,301	19,224	0.48

Table 7: MCTS-RAVE with Havannah-Mate play-out (HM) disabled after $MaxTurns/c$ turns vs. RAVE with random play-out, $t = 5$ seconds

6.2.2 RAVE Play-out

RAVE Play-out is tested with 18,000 play-outs per turn for $p = 0.25, 0.5, 0.75$.

As can be seen in Table 8, MCTS-RAVE with RAVE Play-out increases the win rate by 3% for $p = 0.5$ and $p = 0.75$.

p	0.25	0.5	0.75
Win rate	0.51	*0.53*	*0.53*

Table 8: MCTS-RAVE with RAVE Play-out vs. MCTS-RAVE with random play-out, 18,000 play-outs per turn

However, Table 9 shows that MCTS-RAVE with RAVE Play-out performs weakly when both players run play-outs for $t = 5$ seconds. The reason for that might be that MCTS-RAVE with random play-out is able to run approximately 18,000 play-outs in 5 seconds, whereas MCTS-RAVE with RAVE Play-out manages to run only about 9,000 to 14,500 play-outs depending on how parameter p is set.

p	0.25	0.5	0.75
Win rate	*0.36*	0.29	0.26
# play-outs	14,489	11,908	8,970

Table 9: MCTS-RAVE using RAVE Play-out vs MCTS-RAVE using random play-out, $t = 5$ seconds

6.2.3 Biased Play-out Corner

Biased Play-out Corner is tested for various values of w_c. The results (Table 10) show that biasing the corner cells does not improve the overall playing strength for any w tested.

weight w_c	2	3	5
win rate	0.47	0.42	0.25

Table 10: MCTS-RAVE with Biased Play-out Corner vs. MCTS-RAVE with random play-out

6.2.4 Biased Play-out Proximity

Biased Play-out Proximity is tested for $P_{range} = 1$. Table 11 and Table 12 show the results for various H, w_p and P_{prox}. Biased Play-out Proximity performs weakly for the tested parameters in both standard and alternative mode. On top of the weak results, Biased Play-out Proximity also slows down the play-outs by at least 4%. Though, as the play-out was limited by number of play-outs per turn, not by time, a sub optimal implementation of Biased Play-out Proximity is not the reason for its weak playing strength.

6.3 Progressive History with Havannah-Mate

Both Progressive History and Havannah-Mate increase the chance of winning. In this subsection a Progressive History player with Havannah-Mate play-out is tested

# play-outs	Slow down	H	w_p	Win rate
18,000	0.05	1	3	0.28
18,000	0.05	1	7	0.26
18,000	0.09	2	3	0.21
18,000	0.10	2	7	*0.30*

Table 11: MCTS-RAVE with Biased Play-out Proximity standard mode vs. MCTS-RAVE with random play-out, limited by number of play-outs per turn

# play-outs	Slow down	H	P_{prox}	Win rate
18,000	0.05	1	0.1	0.48
18,000	0.04	1	0.25	*0.51*
18,000	0.06	1	0.5	0.29
18,000	0.05	2	0.1	*0.51*
18,000	0.07	2	0.25	0.50
18,000	0.06	2	0.5	0.34

Table 12: MCTS-RAVE with Biased Play-out Proximity alternative mode vs. MCTS-RAVE with random play-out, limited by number of play-outs

against a regular UCT player without enhancements, therefore $n = 500$ games are played on a board of size $s = 6$, with $C = 1.8$ for the UCT player, and $C = 0.3$, and $W = 10$ for MCTS-PH with Havannah-Mate play-out. Both players run play-outs for $t = 5$ seconds per turn.

The result is that the enhanced player is able to win 76% of the games.

7 Conclusions and Future Research

In this paper we investigated several enhancements for MCTS applied to Havannah. For the selection phase, these are Rapid Action-Value Estimate (RAVE), Progressive History (PH) and Extended RAVE (ER), and for the play-out phase Havannah-Mate, RAVE Play-out, Biased Play-out Corner and Biased Play-out Proximity.

For MCTS-RAVE, we determined its playing strength by matching it against MCTS-UCT. MCTS-RAVE was able to win up to 67% of the games. Therefore we may confirm that this well known technique in computer GO is also efficient in Havannah.

Progressive History was tested by matching MCTS-PH against MCTS-UCT and MCTS-RAVE. Against MCTS-UCT, it was able to win 73% of the games. It also achieved a win rate of up to 61% against MCTS-RAVE. Based on these results, we may conclude that Progressive History is an important enhancement to MCTS based Havannah players.

MCTS-ER was matched against MCTS-RAVE and was able to achieve a win rate of up to 60%, similiar to the win rate of MCTS-PH. But moreover, the results show that MCTS-ER seems to be more affected by sub optimal parameter settings than MCTS-PH. Hence, taking into account that it did not outperform MCTS-PH, we may conclude, that in Havannah Extended RAVE is weaker than Progressive History.

Havannah-mate was the only play-out strategy able to increase the playing strength. If applied only for the first third of the maximum number of turns, it resulted in a win rate of 59% against a player with random play-out. If applied for the complete play-out, the reduction in number of play-outs was too large and it won only 33% of the games. This shows that the number of play-outs is quite relevant.

RAVE Play-out slightly increased the playing strength (53% win rate) when both players were given equal amount of play-outs, but failed to do so when the play-out was limited by time. Though, it might be a viable enhancement if implemented more efficiently.

Biased Play-out Corner and Biased Play-out Proximity both failed to improve the playing strength.

Using Progressive History with Havannah-Mate simultaneously clearly outperformed the regular MCTS player by winning 76% of the games. Even though Lorentz' WANDERER [11] programm is able to win 79% of the games against MCTS-UCT with no enhancements, WANDERER utillizes several more enhancements (among them Havannah-Mate). This indicates that Progressive History is indeed a strong enhancement to MCTS applied to Havannah.

As the experiments show, setting the right parameters is important for getting good results. As future research, finetuning the parameters in a more sophisticated way, e.g. with the Cross-Entropy Method [4] might increase the playing strength. Furthermore, the impact of choosing different T has not been investigated in this paper.

Also, the order the stones are placed in, matters only if a winning connection is achieved. For that reason, nodes that represent the same position but have a permuted history, might be merged to increase efficiency of the information gained by a play-out. For example playing [0,1,2] (i.e. White plays on cell 0, Black on cell 1, White on cell 2) is associated with a different node than playing [2,1,0]. But for considering what move Black should take next, the order in which White played is irrelevant. Consequently these two nodes could be merged to acquire a higher visit count per node with the same number of play-outs.

While our proximity-based play-out strategy did not give good results, a proximity based selection strategy might be viable.

Although Extended RAVE did not outperform Progressive History in Havannah, it was superior to RAVE. So it might be possible that it increases the playing strength in other games, as well. ER could be applied to Chinese Checkers to compare its performance with PH in another game where PH performs well. For games, in which PH does not perform well it would be interesting to test whether ER is viable, as it is partly based on different properties than PH.

References

[1] Brügmann, B. (1993). Monte Carlo Go. Technical report, Physics Department, Syracuse University.

[2] Chaslot, G.M.J-B., Winands, M.H.M., Uiterwijk, J.W.H.M., Herik, H.J. van den, and Bouzy, B. (2008a). Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation*, Vol. 4, No. 3, pp. 343–357.

[3] Chaslot, Guillaume, Bakkes, Sander, Szita, Istvan, and Spronck, Pieter (2008b). Monte-carlo tree search: A new framework for game ai. *AIIDE* (eds. Christian Darken and Michael Mateas), The AAAI Press.

[4] Chaslot, Guillaume, Winands, Mark H. M., Szita, Istvan, and Herik, H. Jaap van den (2008c). Cross-Entropy for Monte-Carlo Tree Search. *ICGA Journal*, Vol. 31, No. 3, pp. 145–156.

[5] Coulom, Rémi (2006). Efficient selectivity and backup operators in monte-carlo tree search. *Computers and Games* (eds. H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. Donkers), Vol. 4630 of *Lecture Notes in Computer Science*, pp. 72–83, Springer.

[6] Freeling, C. (2003). Introducing Havannah. *Abstract Games*, Vol. 14, p. 14.

[7] Gelly, Sylvain and Silver, David (2007). Combining Online and Offline Knowledge in UCT. *ICML* (ed. Zoubin Ghahramani), Vol. 227 of *ACM International Conference Proceeding Series*, pp. 273–280, ACM.

[8] Hart, Timothy and Edwards, Daniel (1963). The Alpha-Beta Heuristic. Technical report, Cambridge, MA, USA.

[9] Joosten, B. (2009). Creating a Havannah Playing Agent. Bachelor's thesis, Department of Knowledge Engineering, Maastricht University, Maastricht, Netherlands.

[10] Kocsis, Levente and Szepesvári, Csaba (2006). Bandit Based Monte-Carlo Planning. *ECML* (eds. Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou), Vol. 4212 of *Lecture Notes in Computer Science*, pp. 282–293, Springer.

[11] Lorentz, R.J. (2010). Improving Monte-Carlo in Havannah. *Computers and Games (CG 2010)*. In press.

[12] Nijssen, J.AM. and Winands, M.H.M. (2010). Enhancements for Multi-Player Monte-Carlo Tree Search. *Computers and Games (CG 2010)*. In press.

[13] Robbins, H. (1952). Some Aspects of the Sequential Design of Experiments. *Bulletin of the American Mathematical Society*, Vol. 58, No. 2, pp. 527–535.

[14] Schaeffer, J. (1989). The History Heuristic and Alpha-Beta Search Enhancements in Practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, pp. 1203–1212.

[15] Sturtevant, Nathan R. (2008). An Analysis of UCT in Multi-player Games. *Computers and Games* (eds. H. Jaap van den Herik, Xinhe Xu, Zongmin Ma, and Mark H. M. Winands), Vol. 5131 of *Lecture Notes in Computer Science*, pp. 37–49, Springer.

[16] Teytaud, Fabien and Teytaud, Olivier (2009). Creating an Upper-Confidence-Tree Program for Havannah. In van den Herik and Spronck [17], pp. 65–74.

[17] Herik, H. Jaap van den and Spronck, Pieter (eds.) (2010). *Advances in Computer Games, 12th International Conference, ACG 2009, Pamplona, Spain, May 11-13, 2009. Revised Papers*, Vol. 6048 of *Lecture Notes in Computer Science*. Springer.

[18] Herik, H. Jaap van den, Uiterwijk, Jos W. H. M., and Rijswijck, Jack van (2002). Games solved: Now and in the future. *Artif. Intell.*, Vol. 134, No. 1-2, pp. 277–311.

[19] Winands, Mark H. M. and Björnsson, Yngvi (2009). Evaluation Function Based Monte-Carlo LOA. In van den Herik and Spronck [17], pp. 33–44.

[20] Winands, Mark H. M., Werf, Erik C. D. van der, Herik, H. Jaap van den, and Uiterwijk, Jos W. H. M. (2004). The Relative History Heuristic. *Computers and Games* (eds. H. Jaap van den Herik, Yngvi Björnsson, and Nathan S. Netanyahu), Vol. 3846 of *Lecture Notes in Computer Science*, pp. 262–272, Springer.