The 9th International Conference on Computers and Games

# Pruning playouts in Monte-Carlo Tree Search for the game of Havannah

Joris Duguépéroux  Ahmad Mazyad
Fabien Teytaud  Julien Dehos

LISIC - Université du Littoral Côte d'Opale

June 2016

# Outline

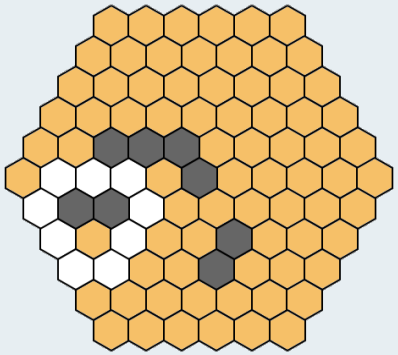
# Outline

# The game of Havannah

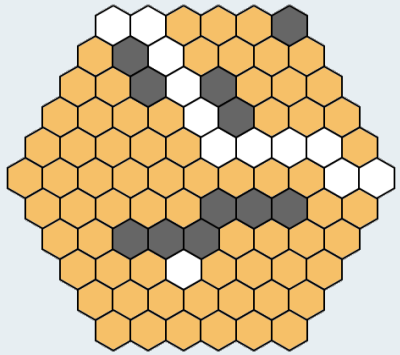
## Presentation

- Invented by Christian Freeling in 1979.
- 2-player connection game.
- Hexagonal board of hexagonal cells.
- At each turn a player has to put a stone in an empty cell.
- To win a player has to realize one of these shapes: fork, bridge, ring.

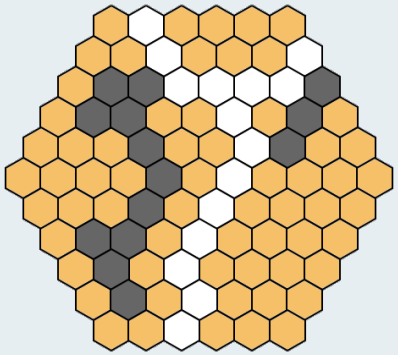
## The winning shapes



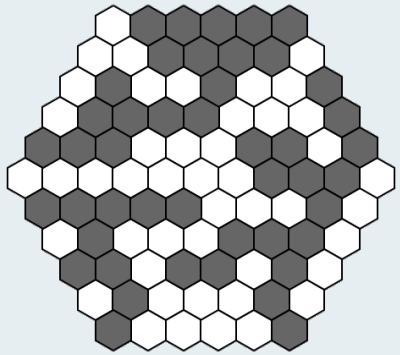
white wins with a ring



white wins with a bridge



white wins with a fork

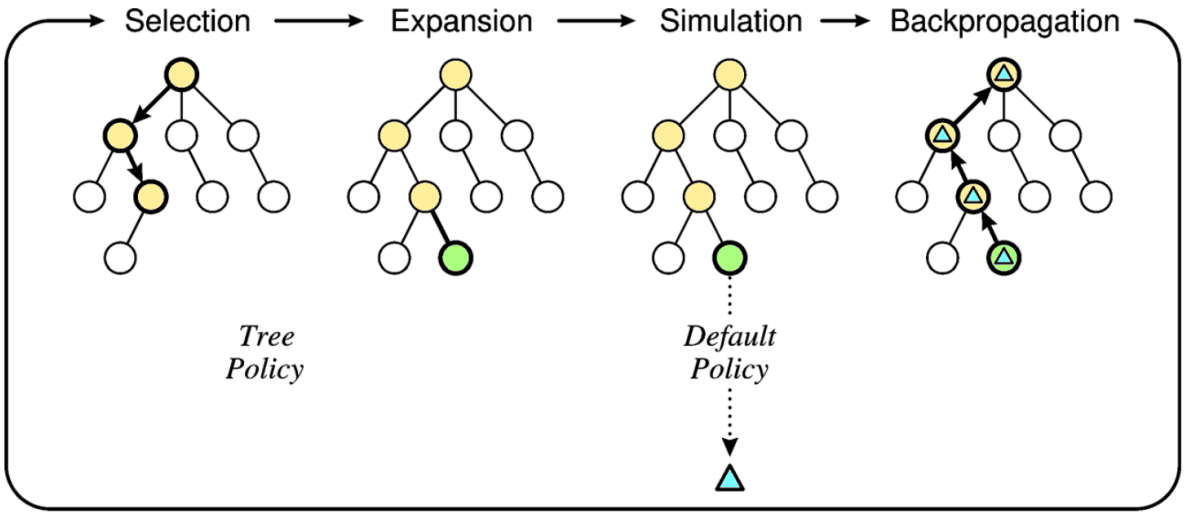


draw

# Outline

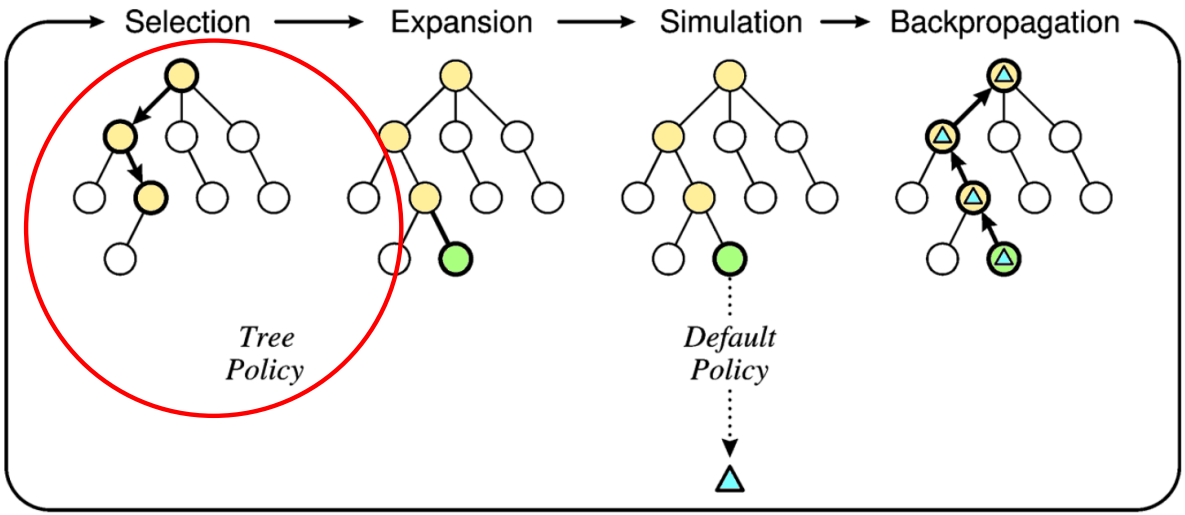
## MCTS principle

- Compute a good move to play.
- Build a tree of the possible future states (tree policy).
- Estimate moves using playouts (default policy).



## Tree policy

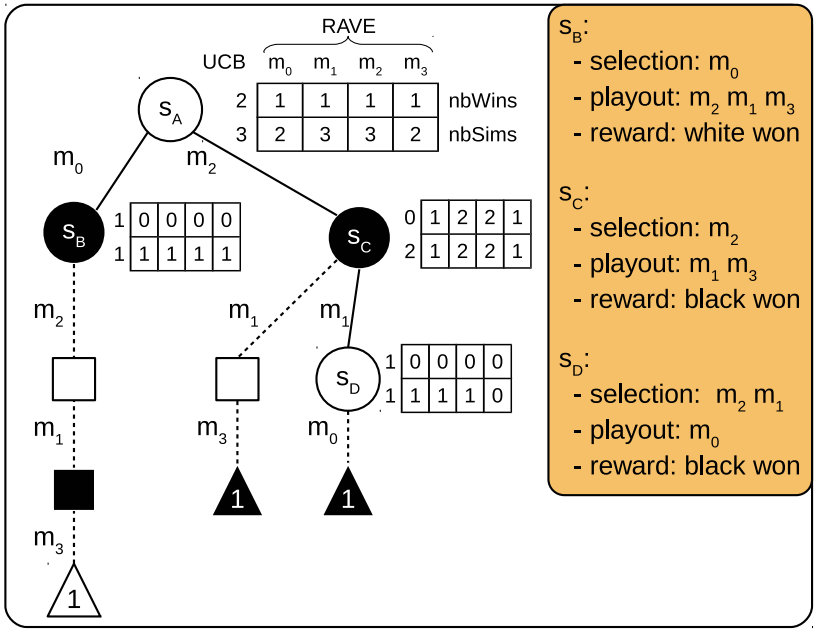
- How to build the tree.
- Classic MCTS: Upper Confidence Bound $\rightarrow \underset{j \in C_{s_1}}{\arg\max} \left[ \frac{w_j}{n_j} + K \sqrt{\frac{\ln(n_{s_1})}{n_j}} \right]$ .
- Improvement with biaising techniques: RAVE...

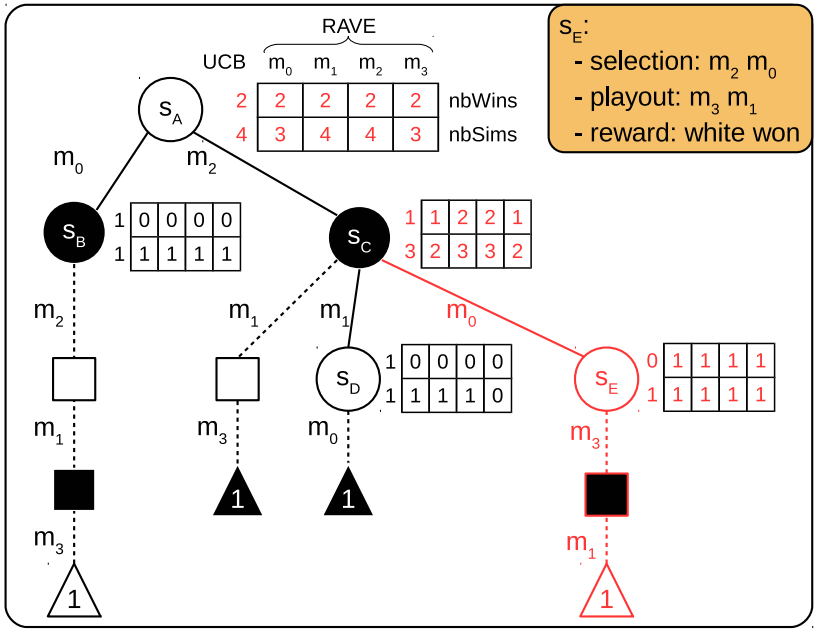


## Tree policy: RAVE

- Rapid Action Value Estimate.
- Gelly and Silver, 2007.
- Select a state using previous playouts

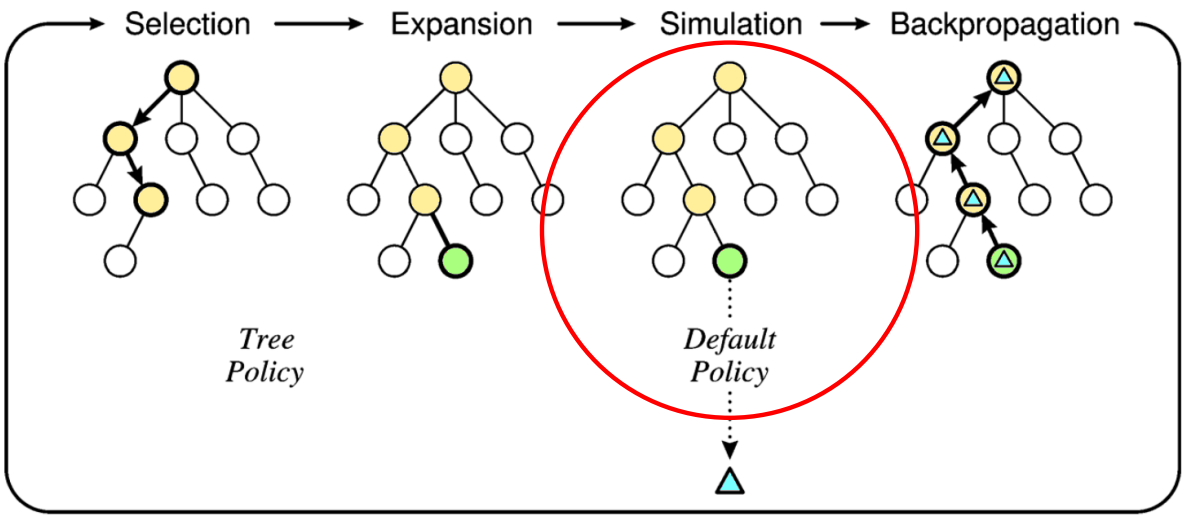
$$\rightarrow \underset{j \in C_{s_1}}{\arg\max} \left[ (1-\beta)\ \frac{w_j}{n_j} + \beta\ \frac{w'_{s_1,j}}{n'_{s_1,j}} + K\ \sqrt{\frac{\ln(n_{s_1})}{n_j}} \right] .$$



## Default policy

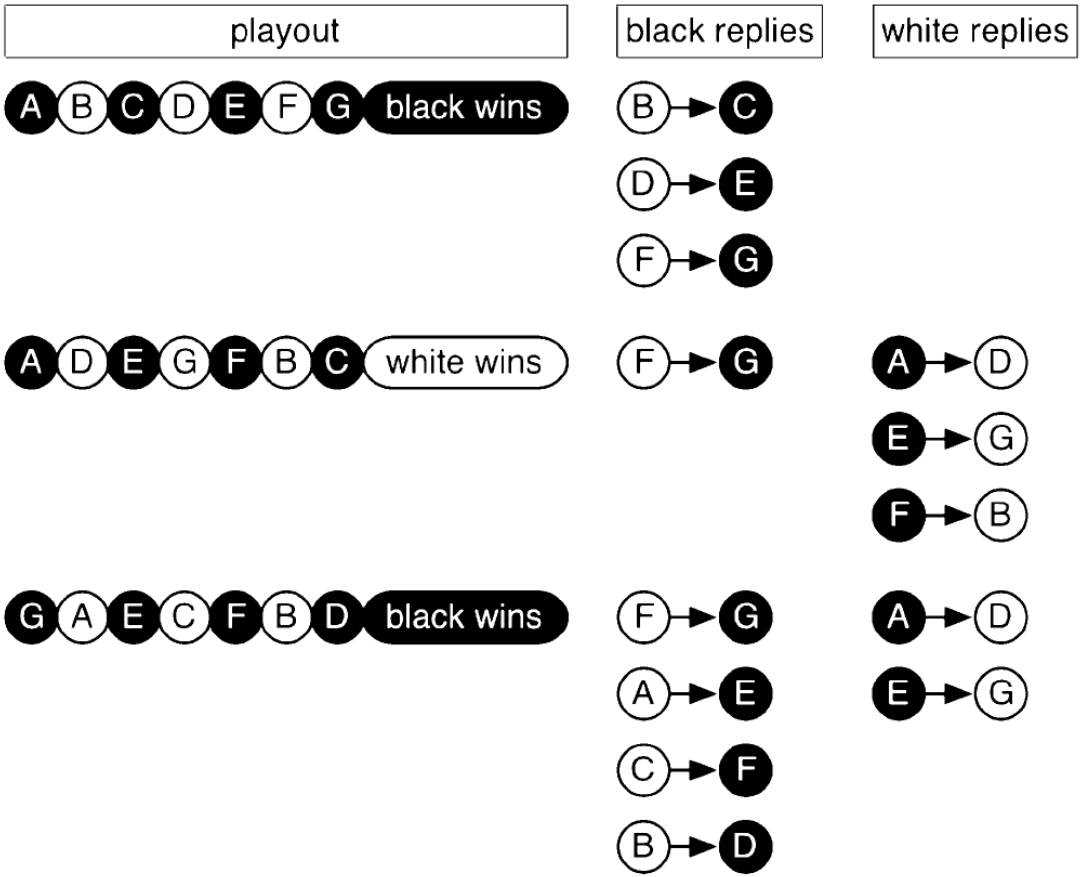
- How to estimate a move.
- Classic MCTS: Monte-Carlo (play random games, compute win rate)
- Improvement with biaising techniques: LGR, PoolRave, N-grams...



## Default policy: LGR

- Last Good Reply.
- Baier and Drake, 2009.
- For each possible moves, remember the lastly played reply if it leads to a win.

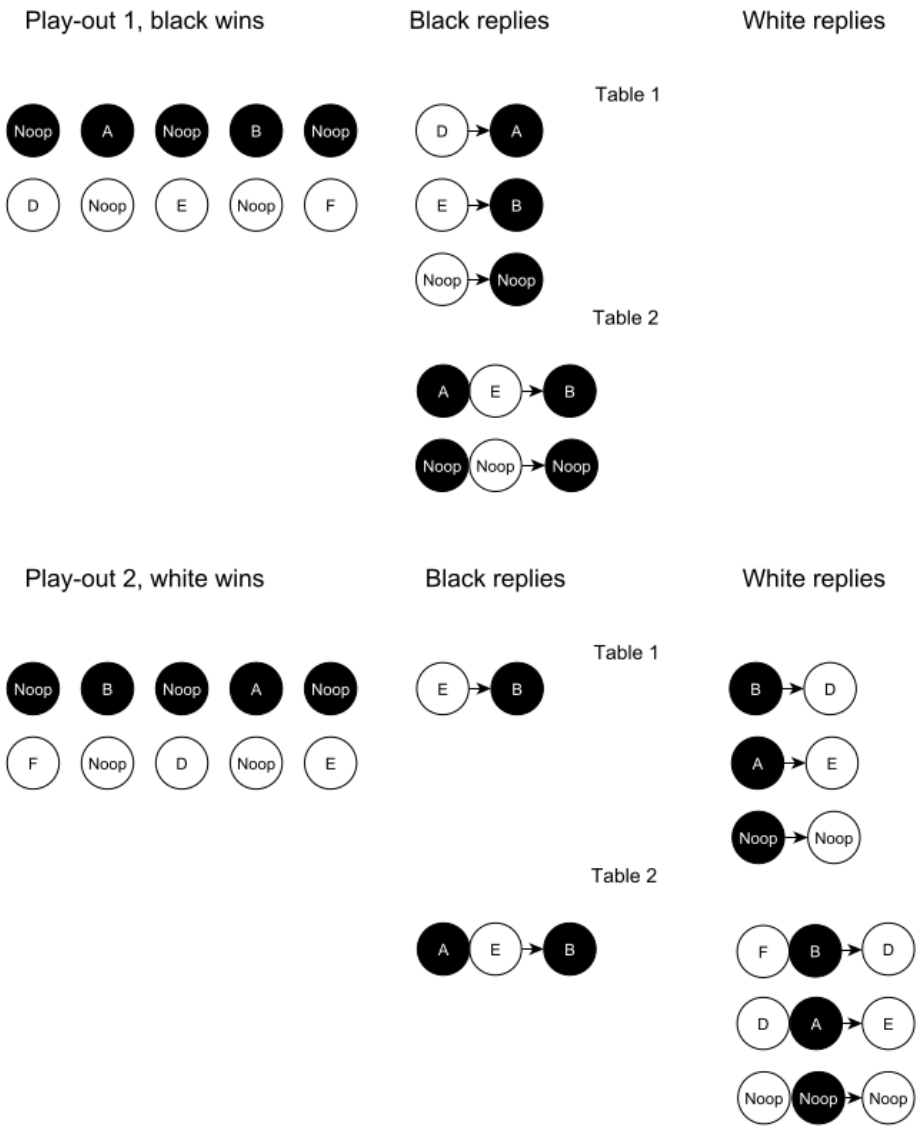
# Monte Carlo Tree Search & beyond



## Default policy: N-grams

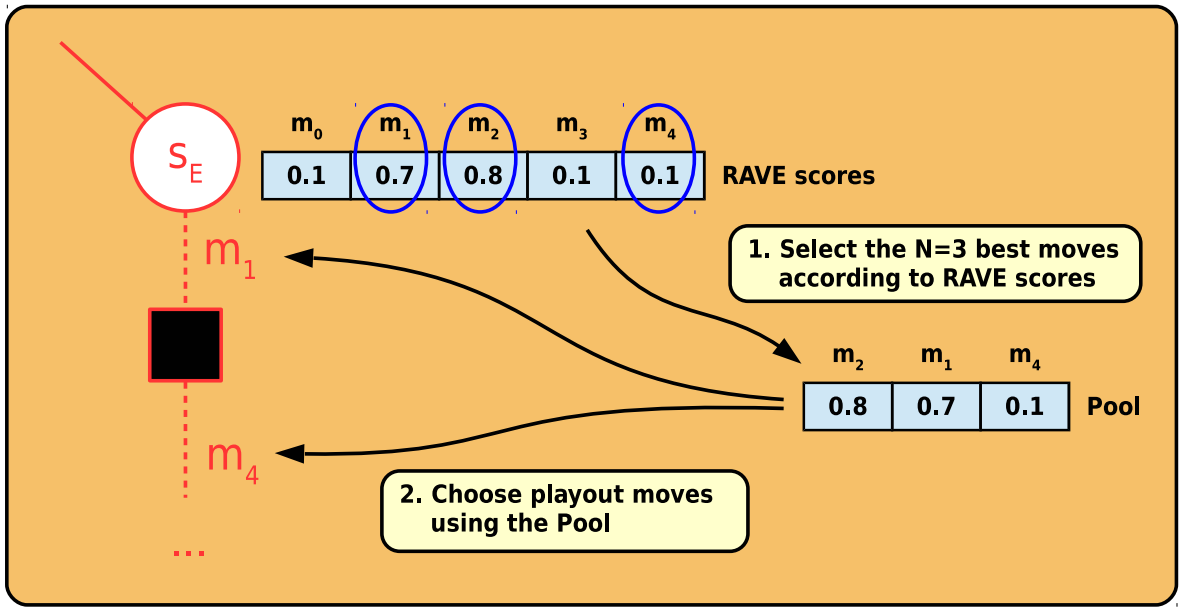
- Tak et al., 2011.
- Remember sequences of N moves which lead to a win.

# Monte Carlo Tree Search & beyond



## Default policy: PoolRave

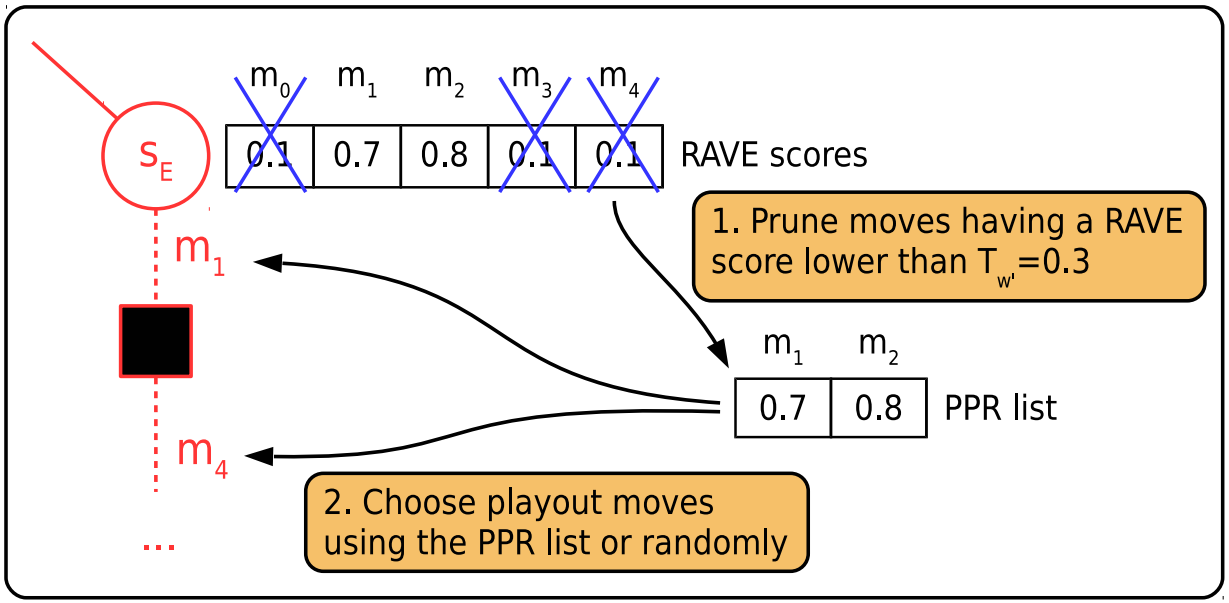
- Rimmel et al., 2011
- Fill a pool of possible moves using RAVE scores.



# Outline

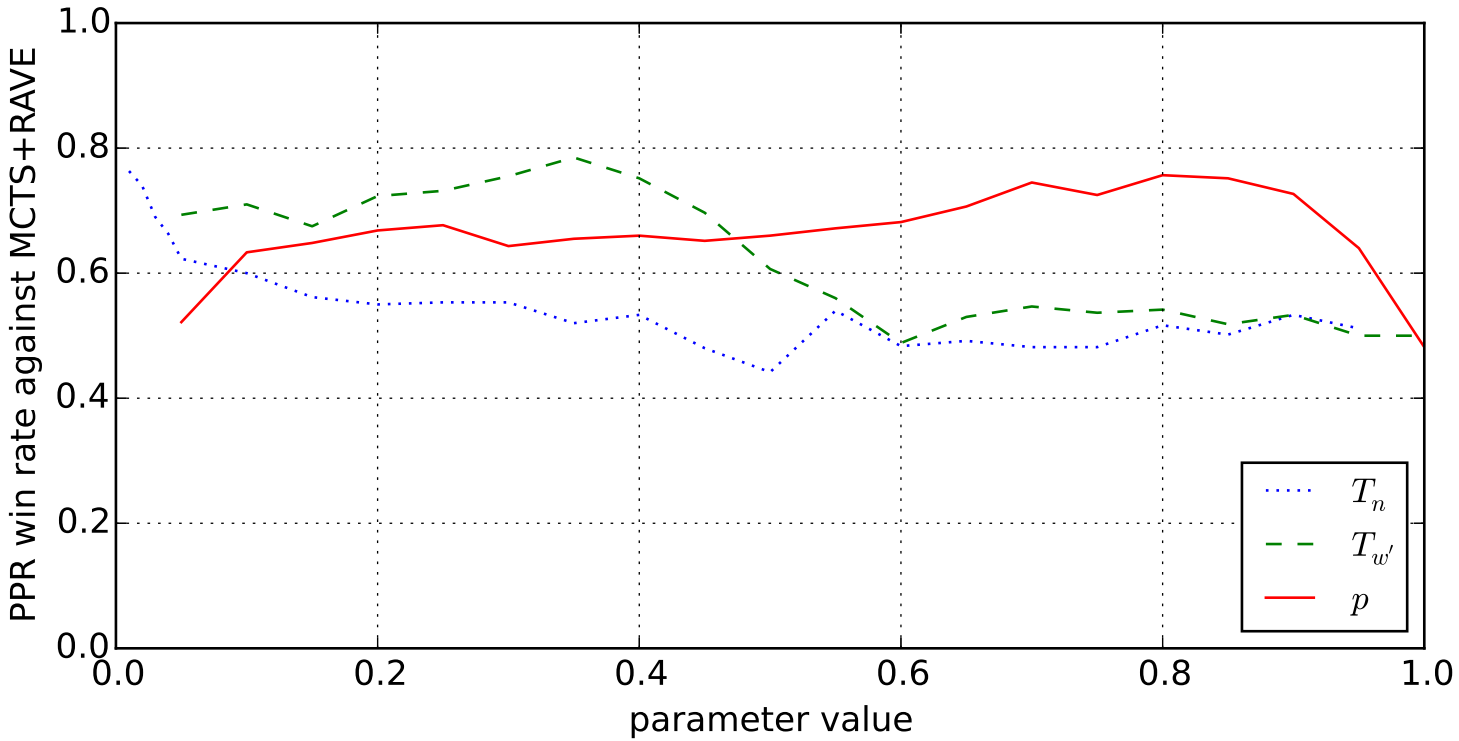
## Pruning Playout with Rave (PPR)

- Biased default policy.
- Prune bad moves, according to RAVE scores.



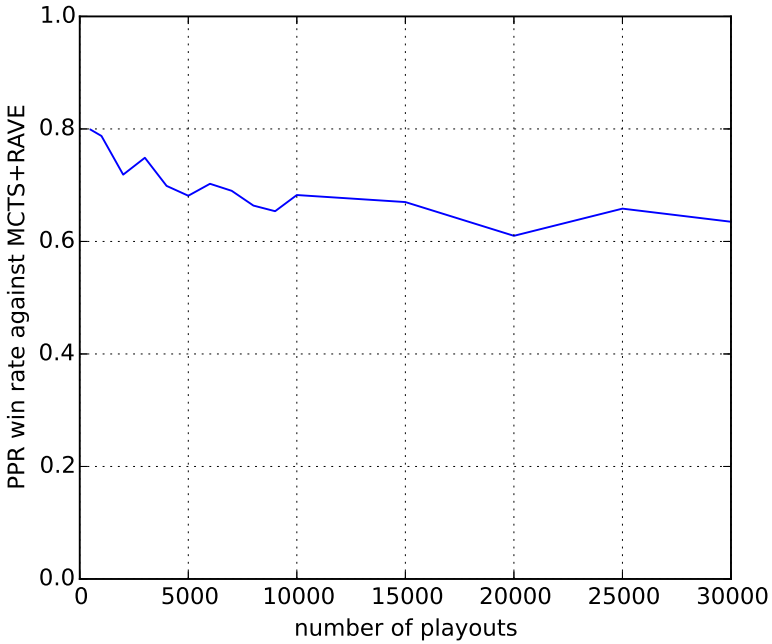
## PPR parameters

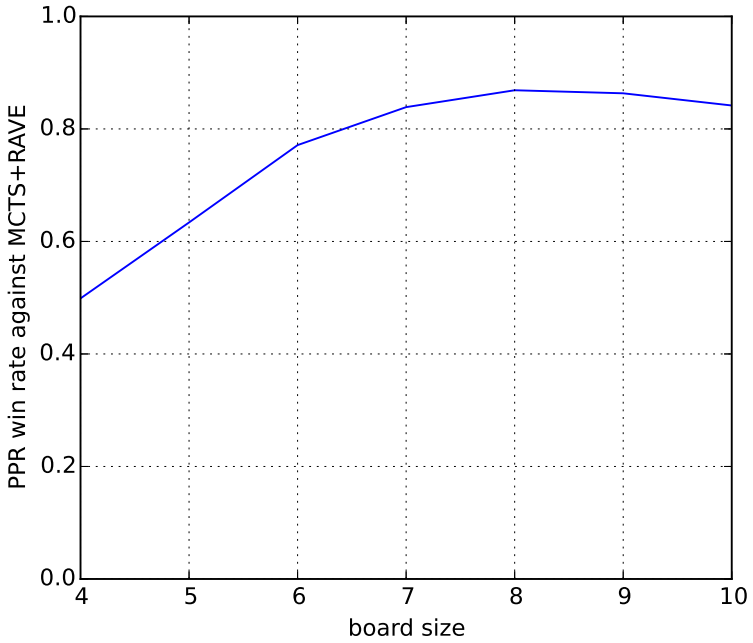
- $T_n$: minimum ratio of playouts for the node to consider for PPR.
- $T_{w'}$: win rate threshold for pruning bad moves.
- $p$: probability for using the PPR list.



## Influence of the number of playouts and of the board size

- PPR is more efficient for "complex" configurations.

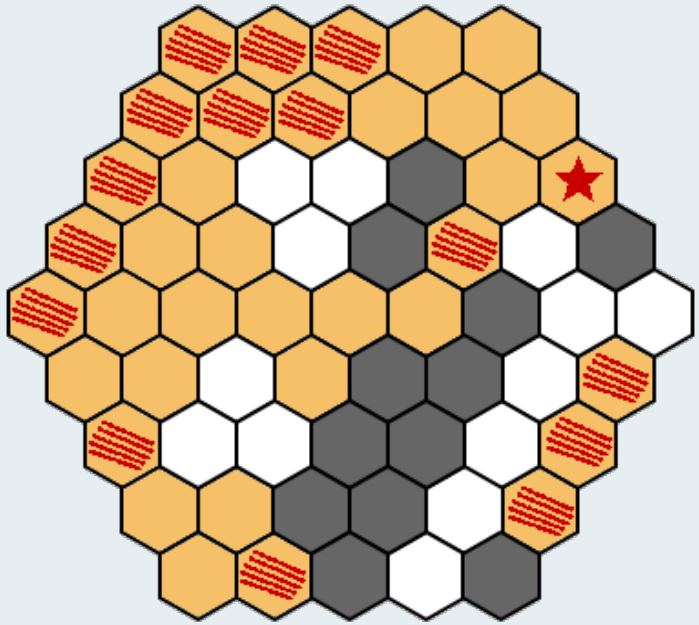


# Proposed method

## Which moves are eventually pruned ?

- after 100k playouts, for black:



# Outline

## Results against other well known improvements (small size)

| size | playouts | player | win rate | std dev |
|---|---|---|---|---|
| 6 | 1,000 | Rave | 74.4% | ±1.78 |
| | | PoolRave | 70.17% | ±1.87 |
| | | LGRF1 | 71.67% | ±1.84 |
| | | Mast | 74.0% | ±1.79 |
| | | Nast2 | 85.0% | ±1.46 |
| | 10,000 | Rave | 63.67% | ±1.96 |
| | | PoolRave | 67.0% | ±1.92 |
| | | LGRF1 | 63.17% | ±1.97 |
| | | Mast | 64.5% | ±1.95 |
| | | Nast2 | 76.5% | ±1.73 |
| | 30,000 | Rave | 66.33% | ±1.92 |
| | | PoolRave | 73.66% | ±1.79 |
| | | LGRF1 | 65.66% | ±1.93 |
| | | Mast | 65.5% | ±1.94 |
| | | Nast2 | 60.5% | ±1.99 |

## Results against other well known improvements (large size)

| size | playouts | player | win rate | std dev |
|---|---|---|---|---|
| 10 | 1,000 | Rave | 86.33% | ±1.40 |
| | | PoolRave | 72.16% | ±1.82 |
| | | LGRF1 | 79.00% | ±1.66 |
| | | Mast | 83.66% | ±1.50 |
| | | Nast2 | 85.50% | ±1.43 |
| | 10,000 | Rave | 79.16% | ±1.65 |
| | | PoolRave | 89.00% | ±1.27 |
| | | LGRF1 | 83.83% | ±1.50 |
| | | Mast | 79.00% | ±1.66 |
| | | Nast2 | 85.16% | ±1.45 |
| | 30,000 | Rave | 75.85% | ±2.13 |
| | | PoolRave | 91.01% | ±1.42 |
| | | LGRF1 | 79.69% | ±2.01 |
| | | Mast | 82.04% | ±1.91 |
| | | Nast2 | 84.08% | ±1.82 |

# Outline

# Conclusion

## Take home message

- Sort of dynamic PoolRave.
- Great results vs other Monte-Carlo improvements (at least 60%).
- Small overall extra cost (if RAVE values are already computed).
- But no such good results on the game of Hex (maybe RAVE is not able to detect "dead areas", if they exist in this game).

# Conclusion

## Future work

- Measure the strongness of PPR with a better bot.
- Test on the game of Go as PoolRave gives good results on this game.

Thank you !

Questions ?