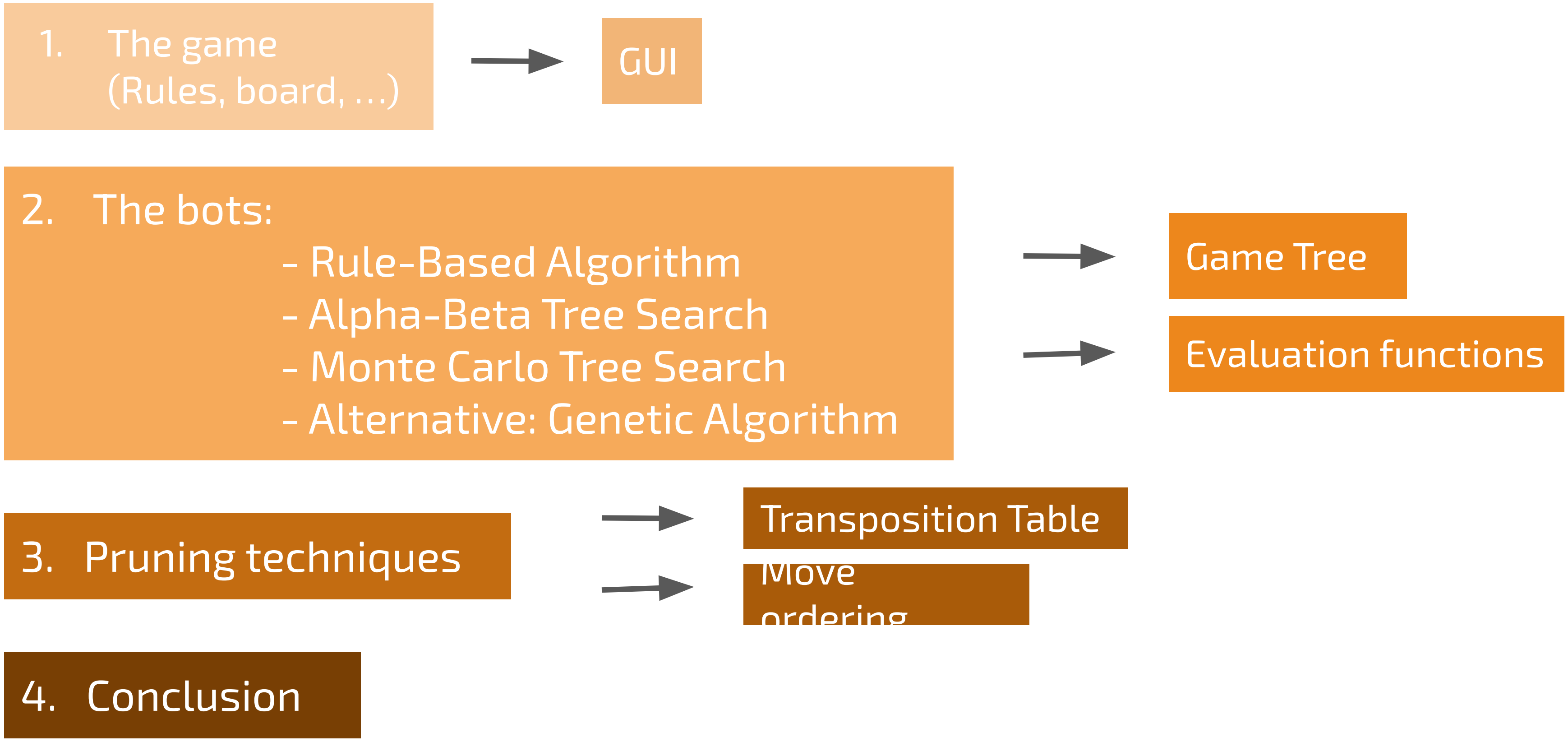
Project 2.1

The Abalone Game

-

Group 5

Table of content



1. The game (Rules, board, …)

GUI

2. The bots:
   - Rule-Based Algorithm
   - Alpha-Beta Tree Search
   - Monte Carlo Tree Search
   - Alternative: Genetic Algorithm

Game Tree

Evaluation functions

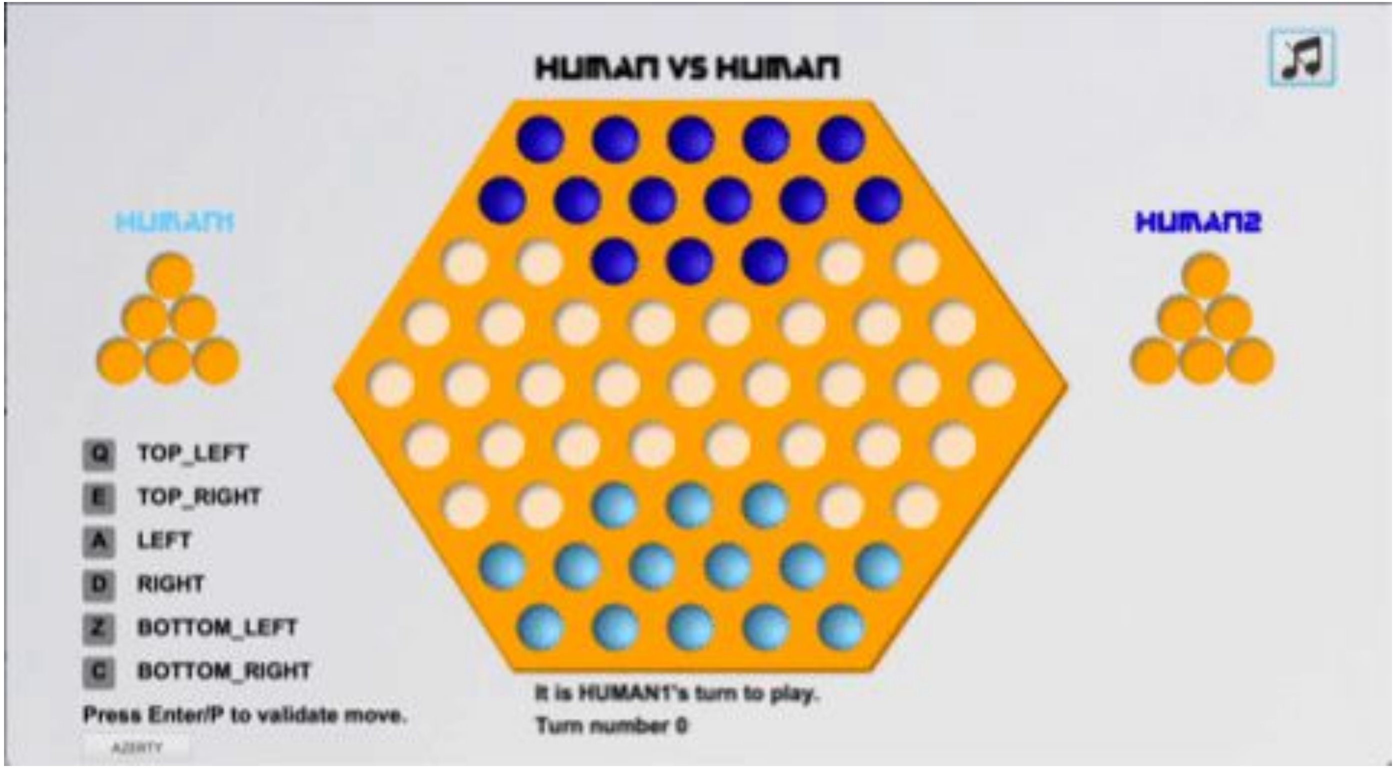
3. Pruning techniques

Transposition Table

Move ordering

4. Conclusion

The Game - Rules



HUMAN VS HUMAN

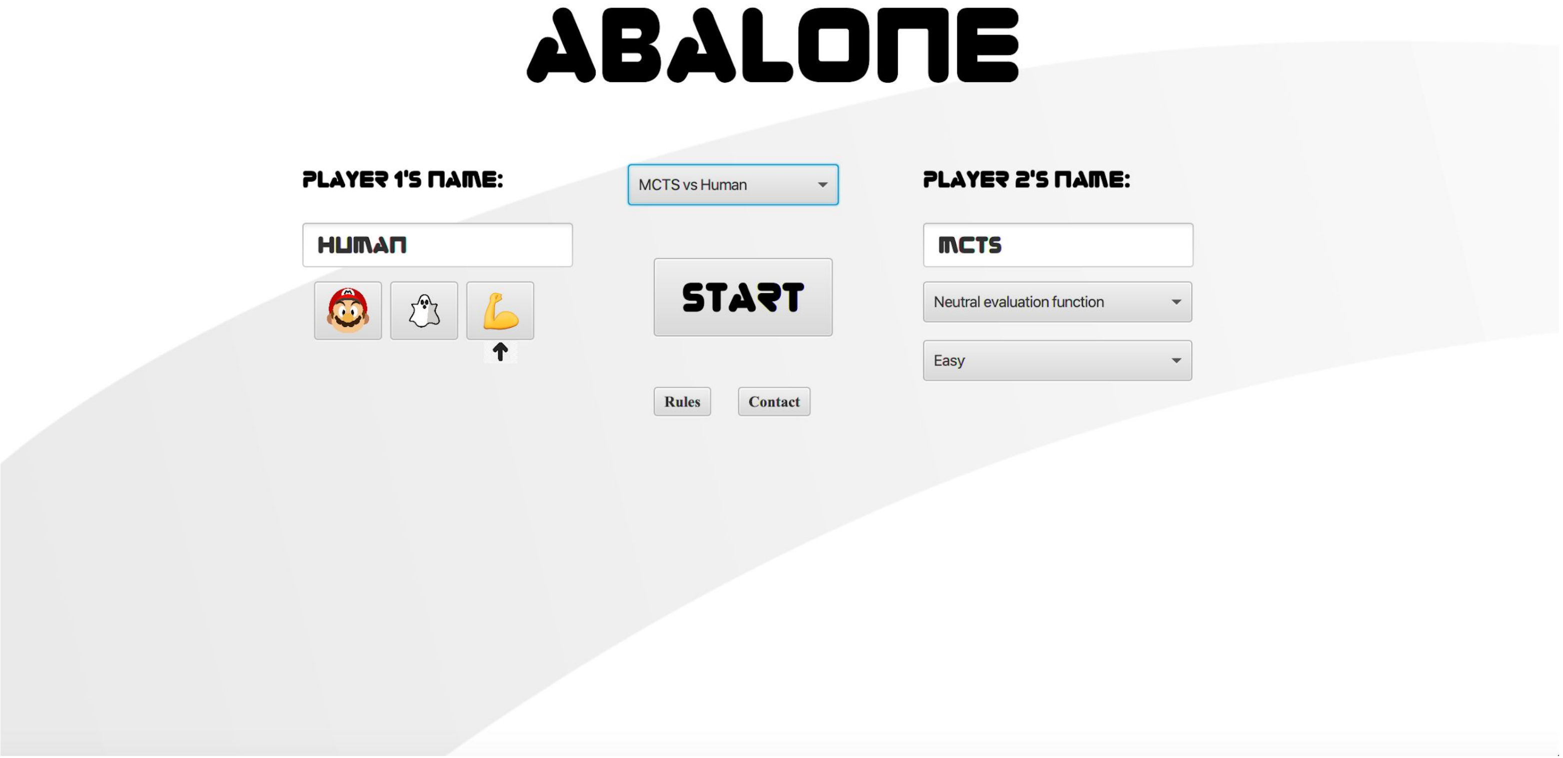
Q	TOP_LEFT
E	TOP_RIGHT
A	LEFT
D	RIGHT
Z	BOTTOM_LEFT
C	BOTTOM_RIGHT

Press Enter/P to validate move.

It is HUMAN1's turn to play.

Turn number 0

Our GUI



ABALONE

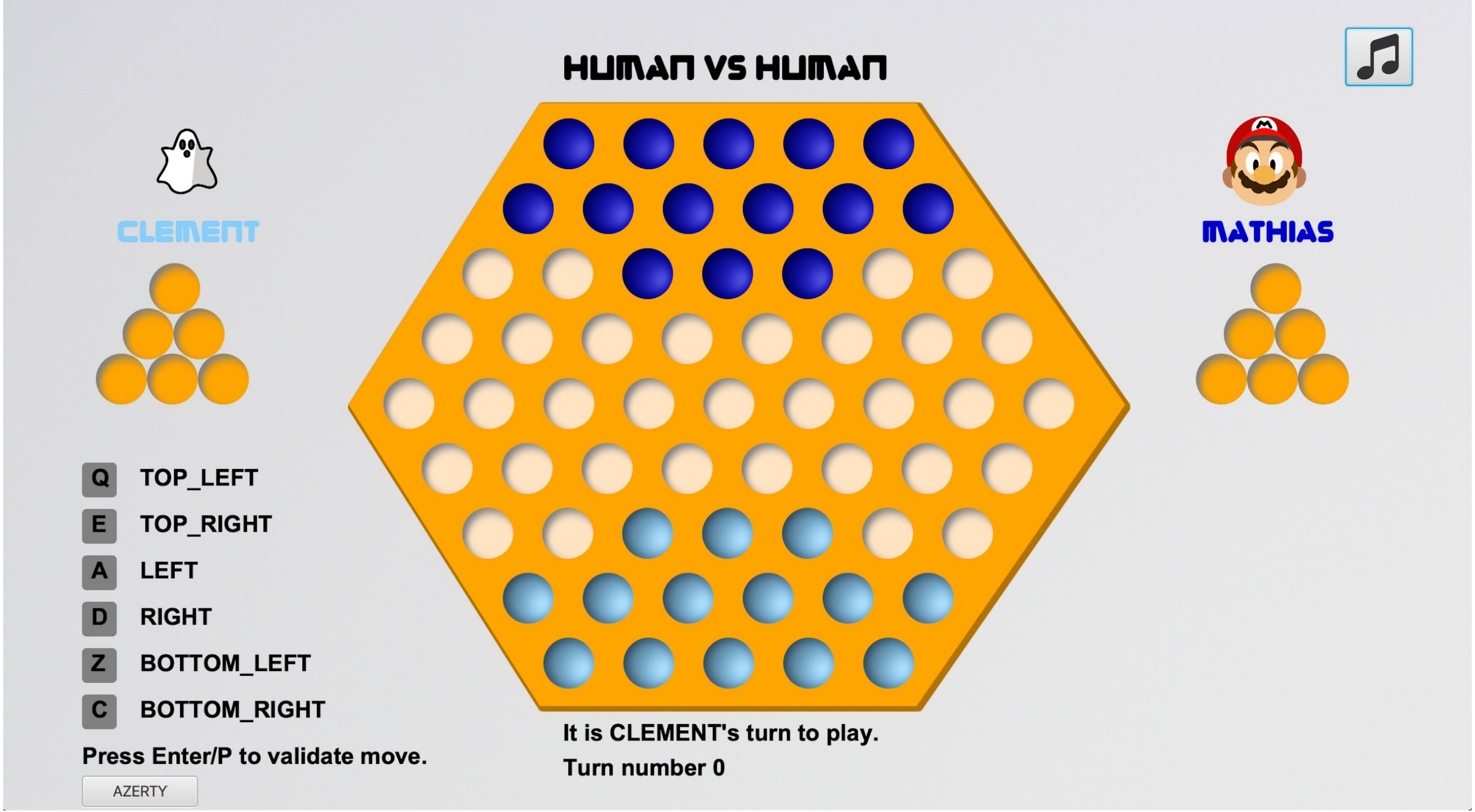
PLAYER 1'S NAME: HUMAN

MCTS vs Human

PLAYER 2'S NAME: MCTS

Neutral evaluation function

Easy



HUMAN VS HUMAN

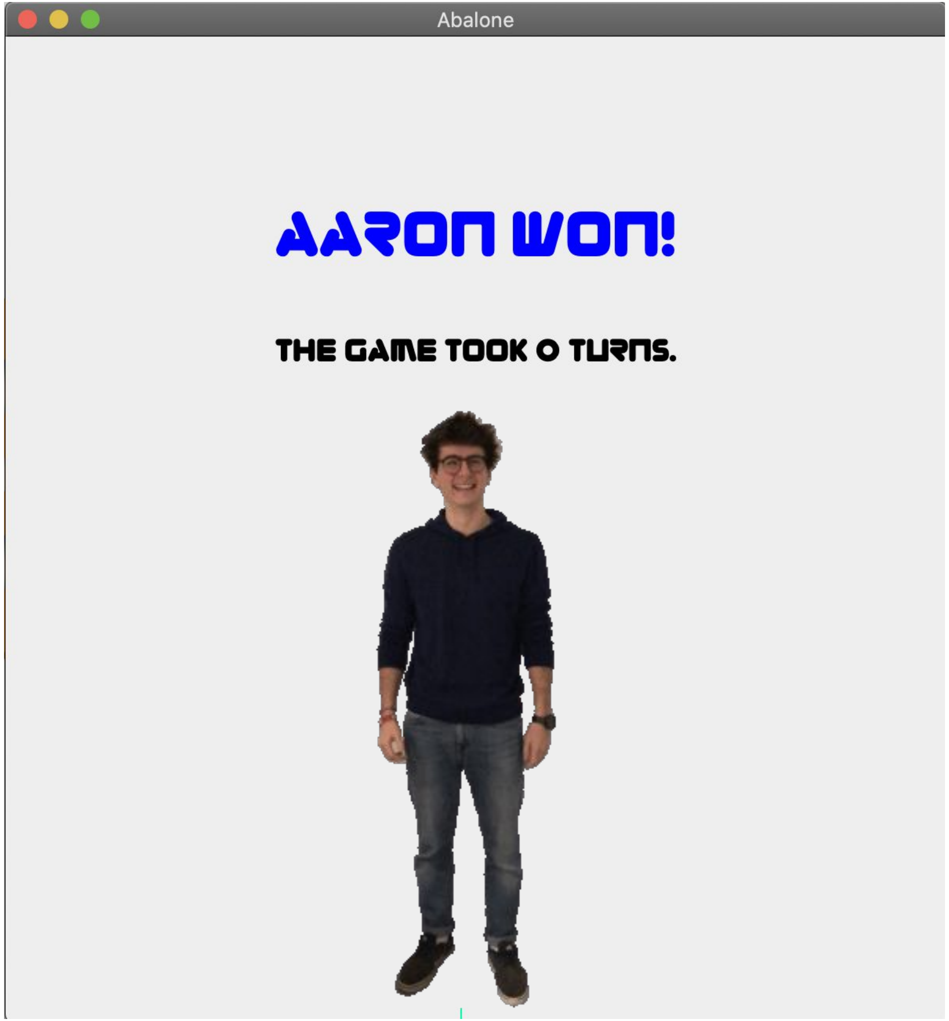
Q	TOP_LEFT
E	TOP_RIGHT
A	LEFT
D	RIGHT
Z	BOTTOM_LEFT
C	BOTTOM_RIGHT

Press Enter/P to validate move.

It is CLEMENT's turn to play.

Turn number 0

Win Page



AARON WON!

THE GAME TOOK 0 TURNS.

Rule-Based Algorithm

```
if there is a sumito move:
    pick random sumito move;
else:
    if there is a pushing move:
        pick random pushing move;
    else:
        if there is a triple move:
            pick random triple move;
        else:
            if there is a double move:
                pick random double move;
            else:
                pick random single move;
```

Sumito move:

any move that ejects an opponent's marble

Pushing move:

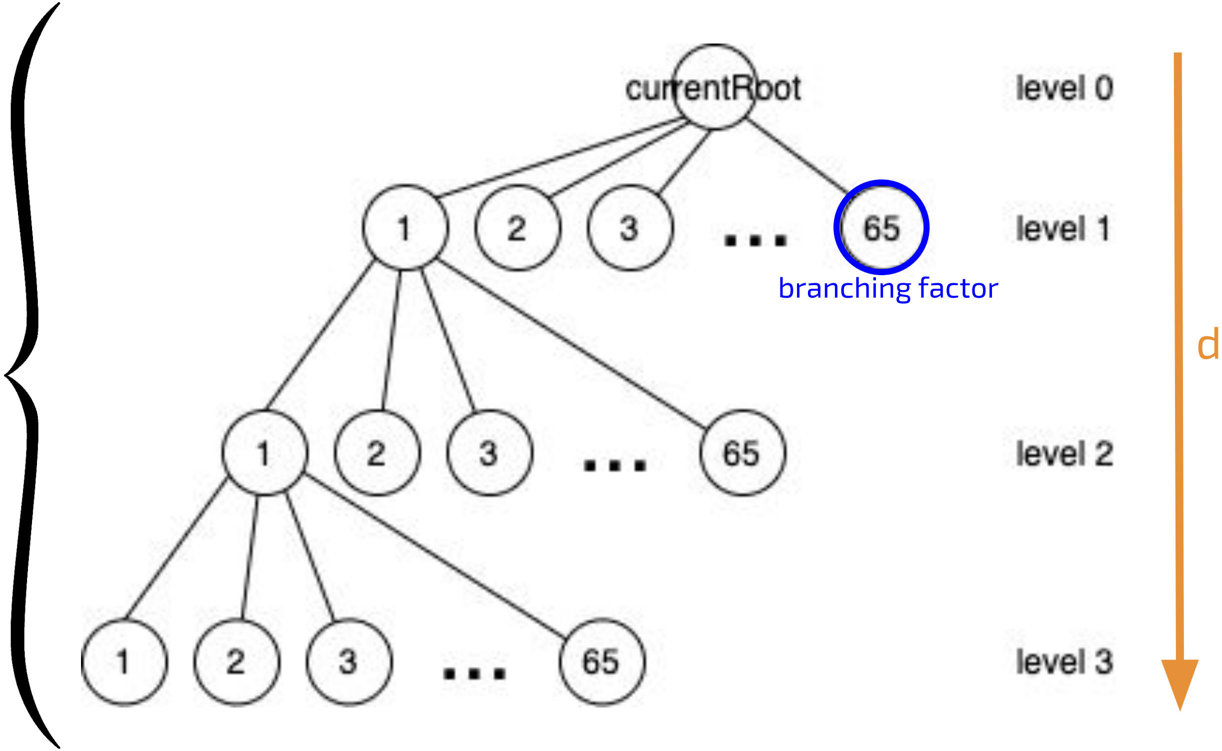
any move that displace an opponent's marble

Super fast at playing but not super performant...

Game Tree Structure

used by ABTS

- creation in BFS : O(b^d)
- visit in DFS



IMPOSSIBLE TO COMPUTE ENTIRE GAME TREE !

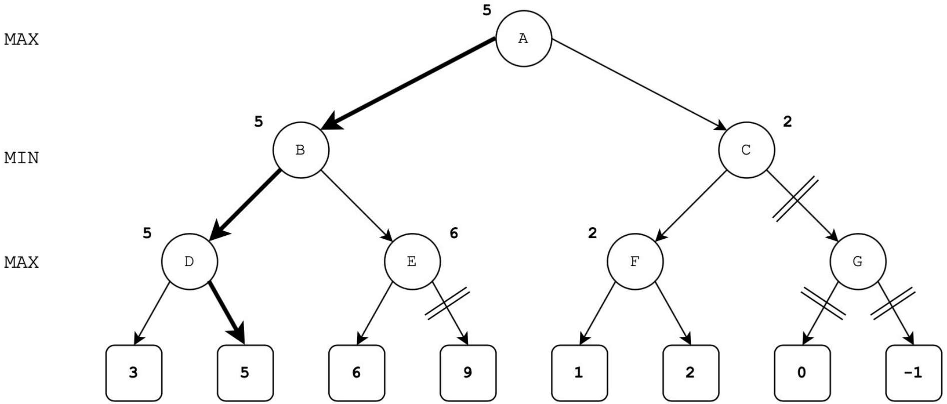
Alpha-Beta Tree Search

Minimax

- Min player and Max player
- Evaluate positions
- Costs a lot of computational time
- Complexity of O(M^D).

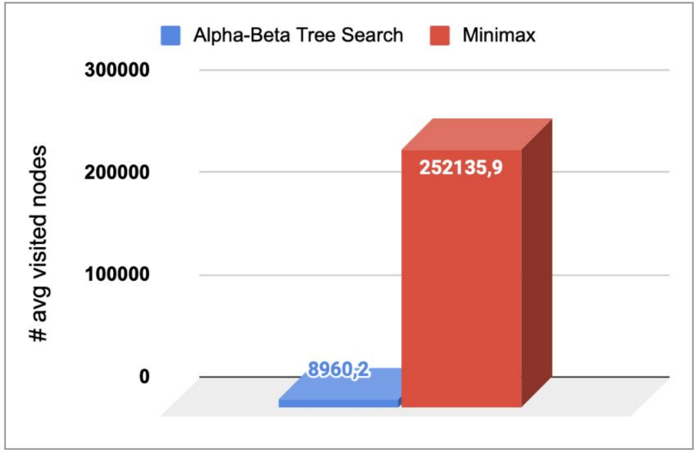
Alpha-Beta pruning

- Will result in the same outcome
- Prunes nodes not worth checking
- Allows the AI player to search at depth 3 in reasonable time
- improves with a better move ordering

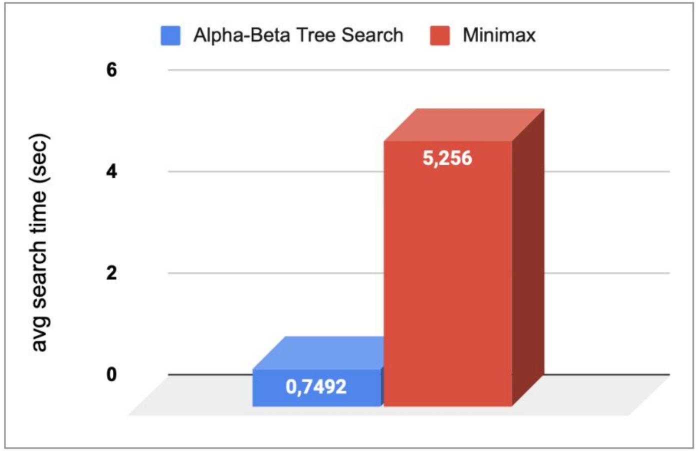


→ Experiment

Alpha-Beta pruning vs MiniMax



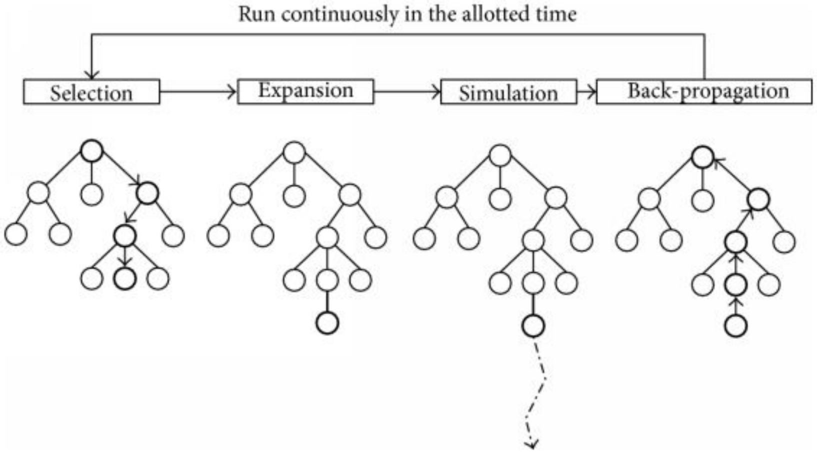
- Huge investigated nodes difference
- Up to 96% saved nodes



- Search algorithm runs faster
- 4.5sec saved on average

Monte-Carlo Tree Search

- **Selection**
  - Start with the root
  - Select the best child until a leaf is reached
- **Expansion**
  - Expands the tree with current node children
- **Simulation**
  - Select a random child
  - Simulate it and get the results
- **Back-propagation**
  - Back-propagate the score till the root

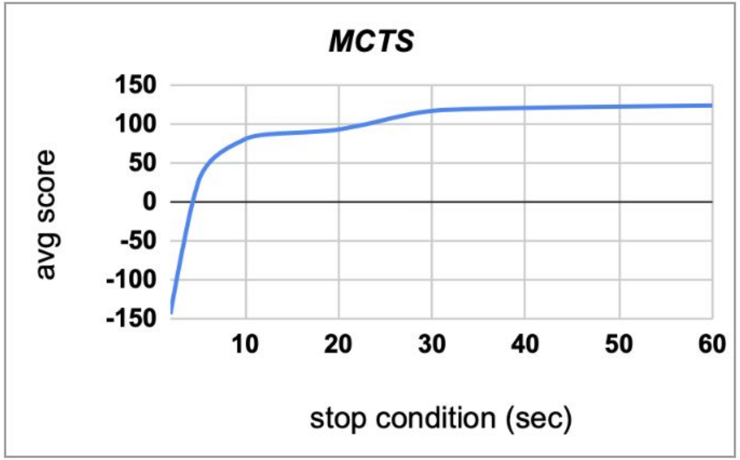


→ Repeat until **stop condition** is met, then output the best move

→ Fixed amount of time

→ Experiment

MCTS stop condition



- Efficiency threshold at 10 sec
- Inefficient below it
- Constant efficiency above

Evaluation Functions

...all based on heuristics

Neutral

- Most efficient

- Weights adapting

- From offensive to defensive (and vice versa)

Offensive

- Focuses on ejecting

- Risked approach

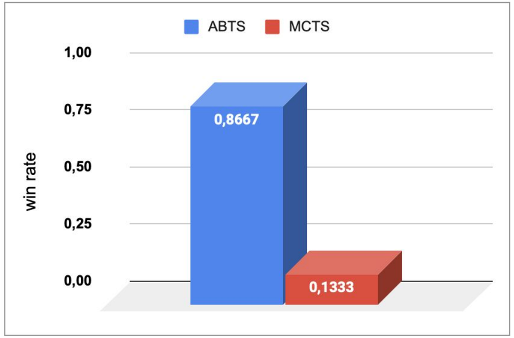
Defensive

- Focuses on consolidating

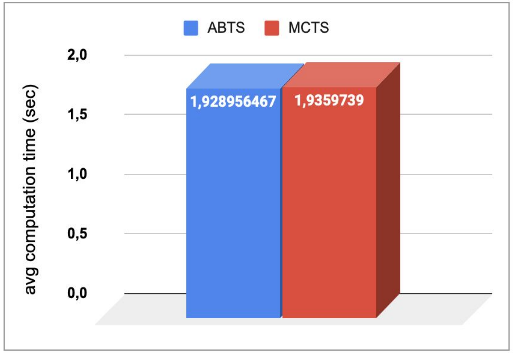
- Safed approach

→ Experiment

ABTS vs MCTS

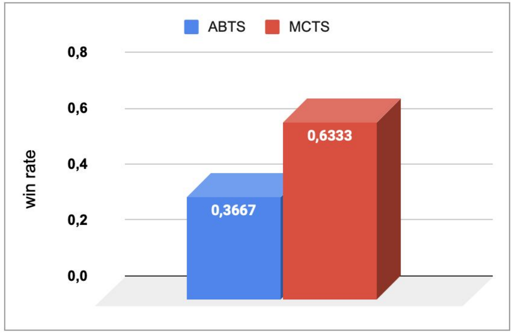


→ ABTS better win rate

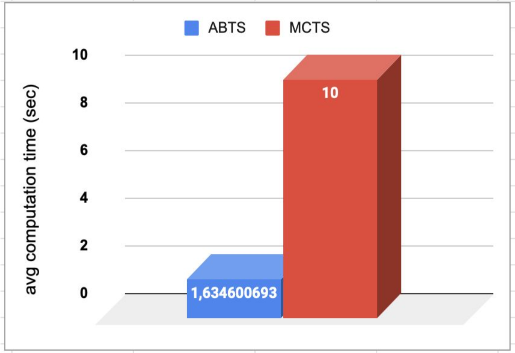


100% fairness

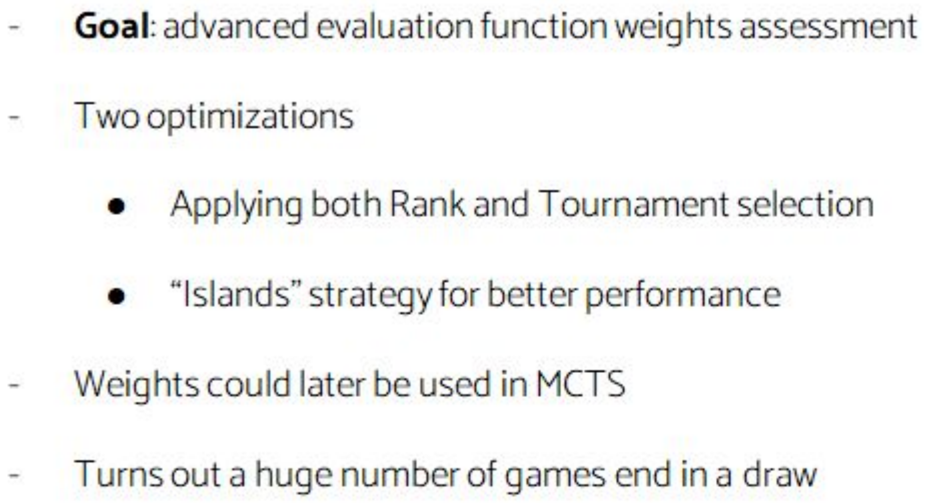
MCTS better win rate →



No fairness



Genetic Algorithm



- **Goal**: advanced evaluation function weights assessment
- Two optimizations
  - Applying both Rank and Tournament selection
  - “Islands” strategy for better performance
- Weights could later be used in MCTS
- Turns out a huge number of games end in a draw

Transposition Table

- Transposition Table = Hash Table
- Hash Function f(x) → unique key
- Key → index of the table
- Table contains already computed node score
- Transposed node reused previous evaluation

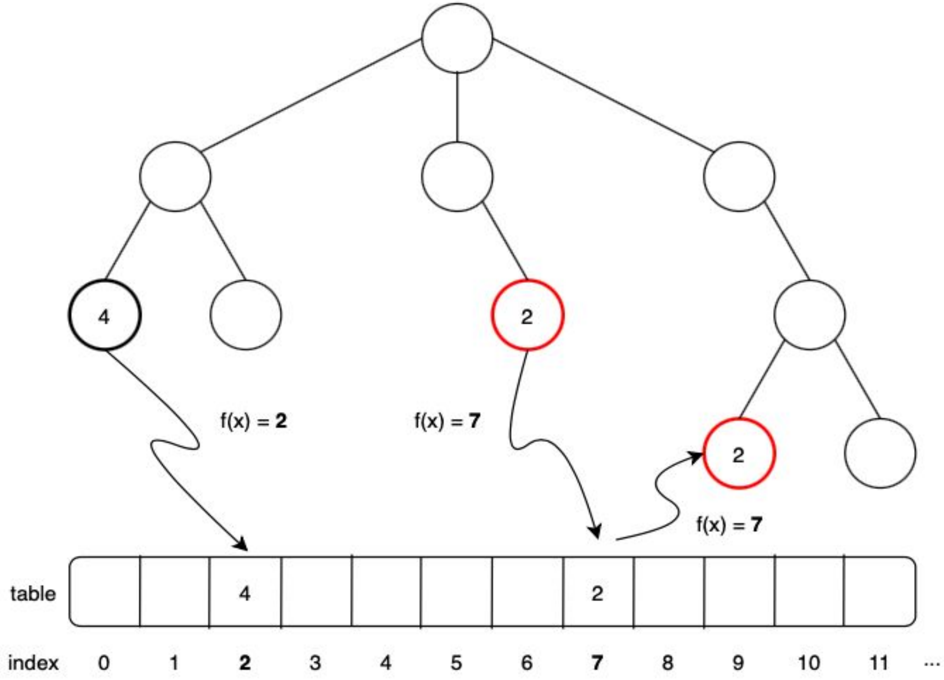
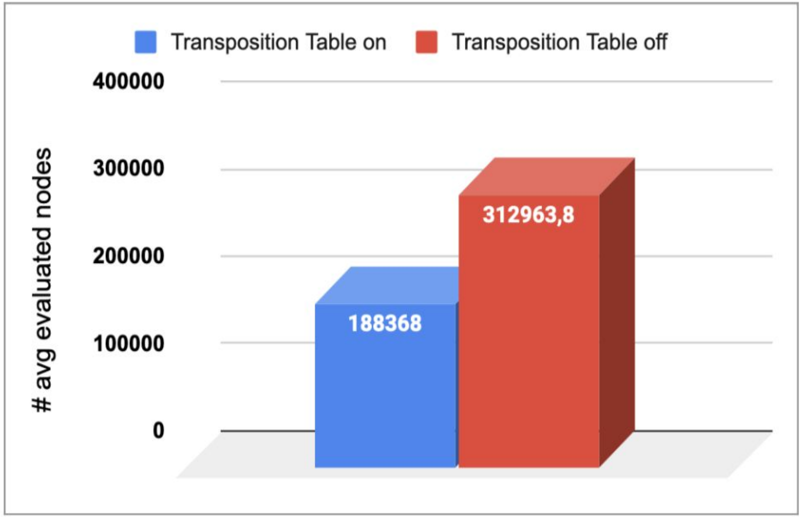


table			4					2					
index	0	1	**2**	3	4	5	6	**7**	8	9	10	11	...

→ Experiment

Transposition Table ON vs OFF



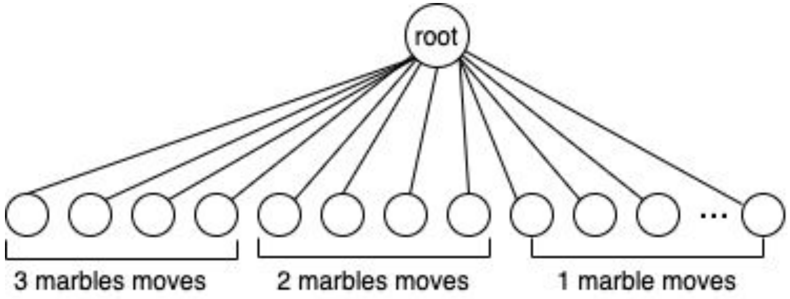
→

- Almost half of evaluated nodes saved
- Computational gain

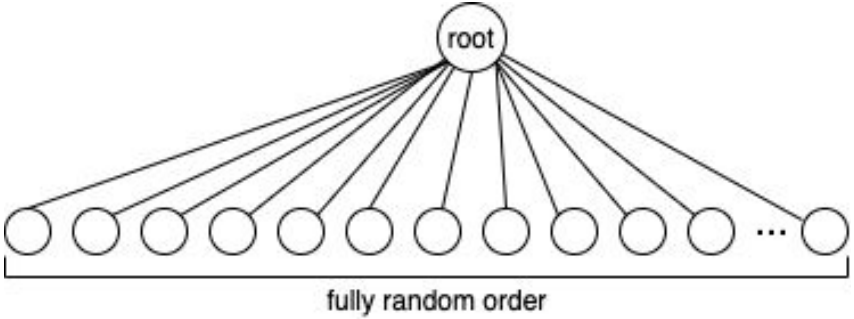
Move ordering

- Improves alpha-beta pruning
- Better moves first
- Three different orderings tested:

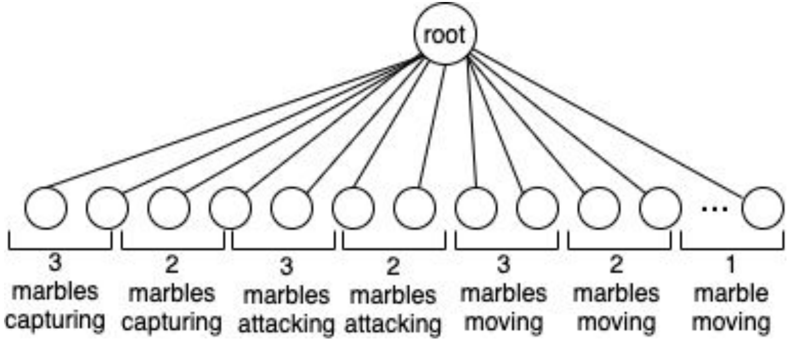
Simple ordering



Random ordering

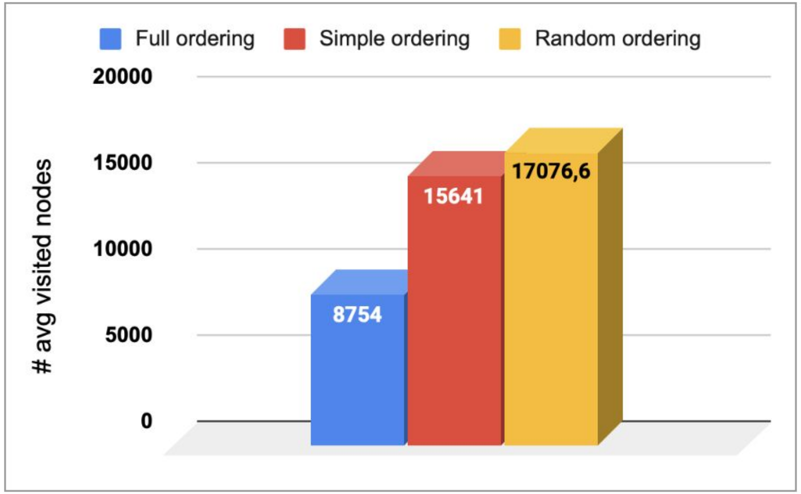


Full ordering



→ Experiment

Move ordering



- Full ordering greatly outperforms the other two orderings
- Simple ordering still outperforms random ordering.
- Prioritizing capturing and pushing is best.

Conclusion

Bots

- Rule Based
- Monte-Carlo Tree Search
- Alpha-Beta Tree Search

Evaluation functions

- Neutral
- Offensive
- Defensive

Pruning techniques

- Move ordering
- Transposition table

THANK YOU FOR LISTENING!