Fanorona

Adversarial Search Methods

Cátia Teixeira
Rojan Aslani

February 2023

Faculty of Engineering of University of Porto
Master's in Data Science and Engineering

Artificial Intelligence course
Professor: Luís Paulo Reis

Contents

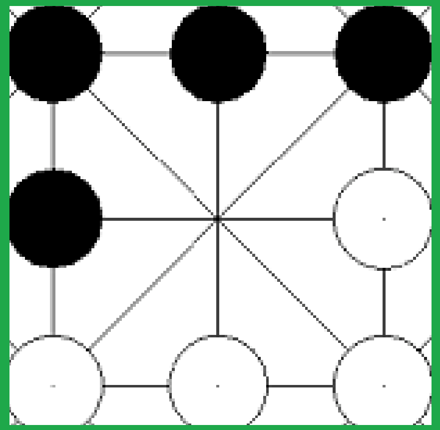
- Introduction
- Methods
- Results and Discussion
- Conclusions and Future Works
- Demo

Introduction

What is Fanorona?

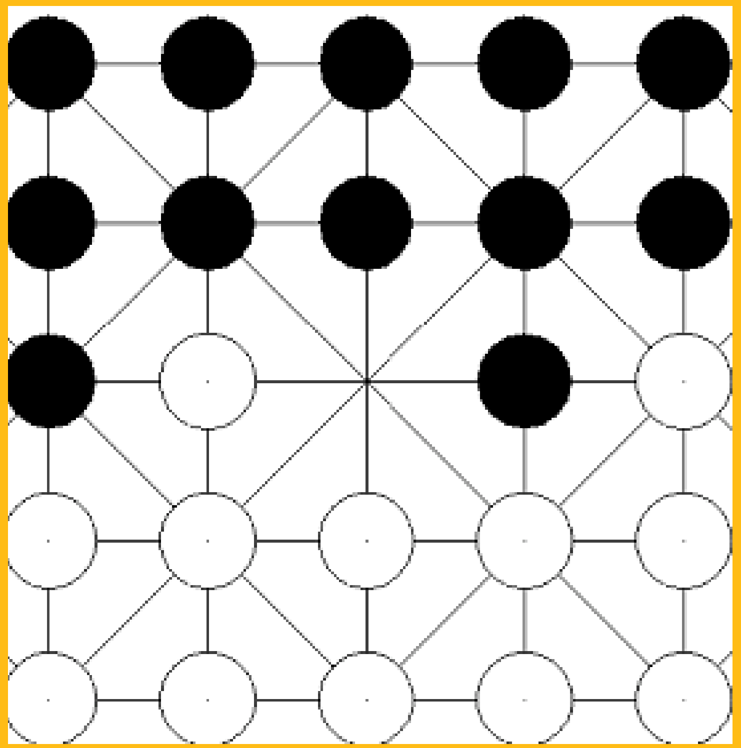
Malagasy national board game.

Board Sizes



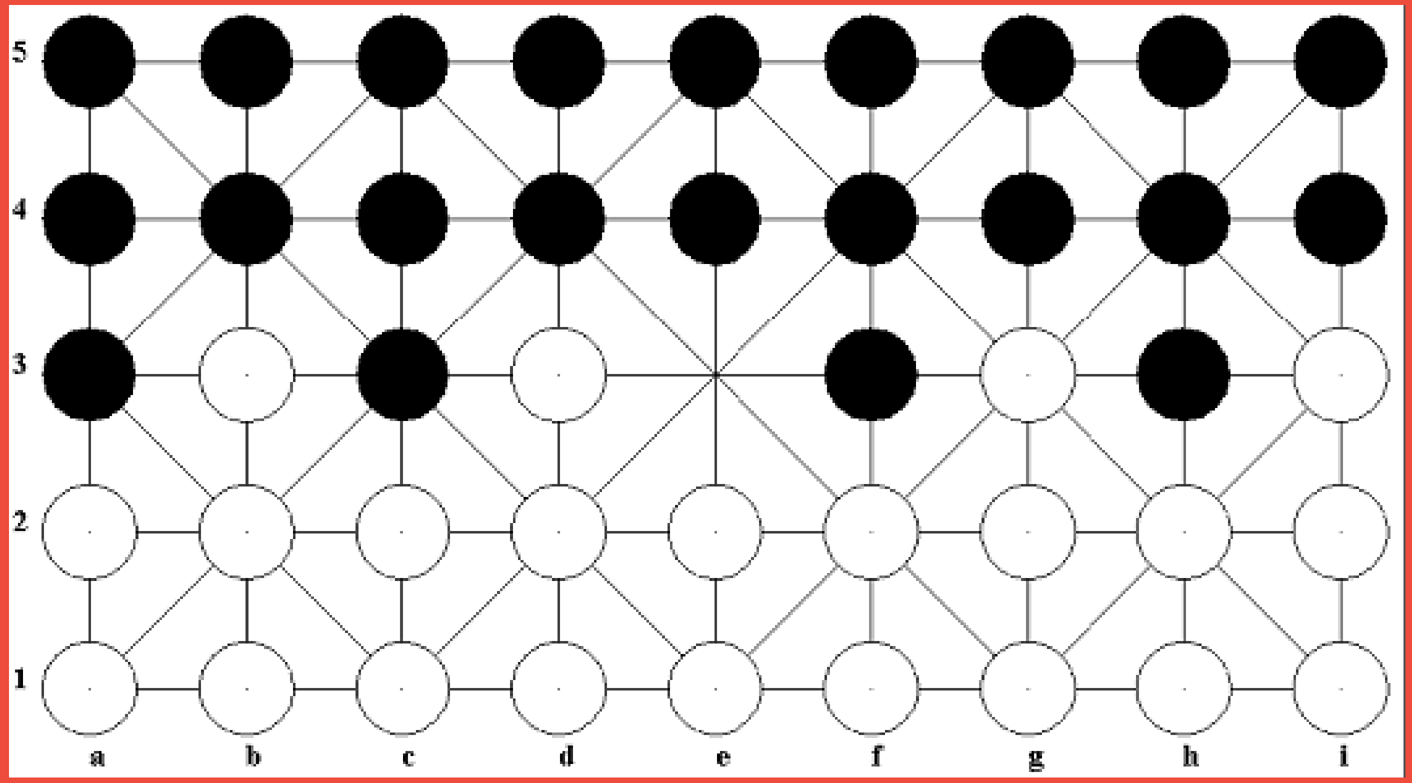
3x3

Fanoron-Telo



5x5

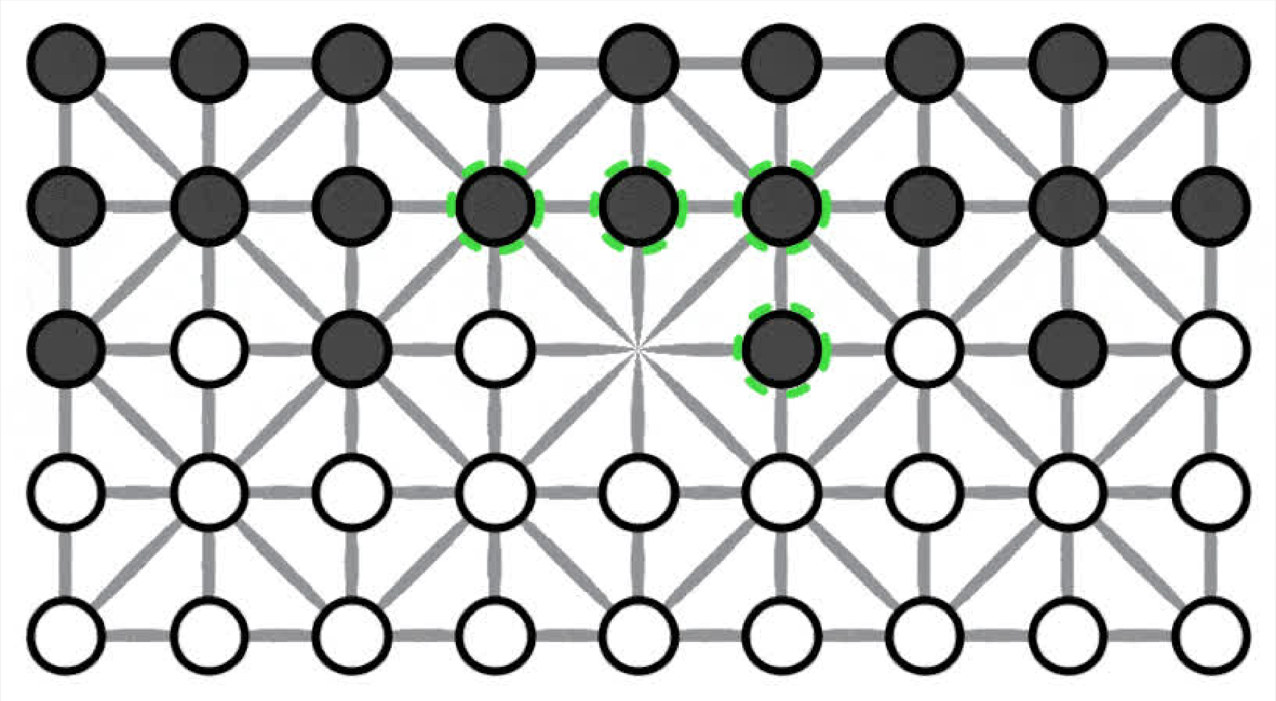
Fanoron-Dimy



9x5

Fanoron-Tsivy

Rules



- White plays first.
- Stones move along the lines.
- Capture happens when the player moves a stone in the same line as the opponent stone adjacent to it.
- In captures, all the stones in the continuous line are captured.
- Capture the opponent's stone by **approach** or **withdrawal**.
- If, after a capture, another capture is available, the player may continue to move.
- Win/Lose/Draw

State of the art

- Accessible
- Deterministic
- Non Episodic
- Static
- Discrete

- Algorithms implemented
  - MINIMAX
  - Monte-Carlo Tree Search (MCTS)
  - many others.

- Until now, Fanorona **has never been strongly solved.**

Methods

Structure of the game

main.py

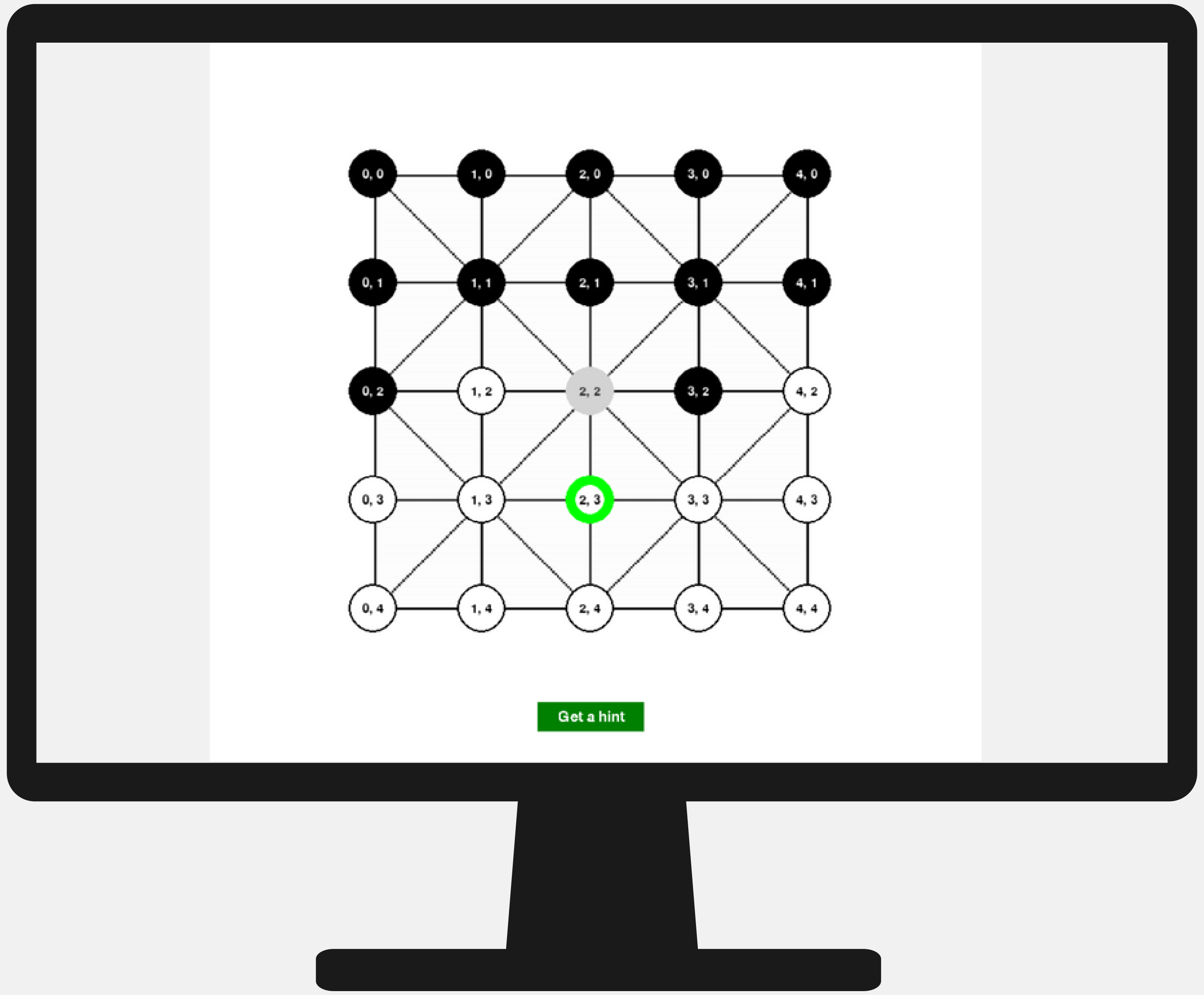
player.py

game.py

board.py

stats.py

Possible game modes



Choose player 1 (White tokens)

Human | Minimax | Minimax_AlphaBeta | Monte_Carlo_TS

Choose player 2 (Black tokens)

Human | Minimax | Minimax_AlphaBeta | Monte_Carlo_TS

In which board size would you like to play with?

Fanoron-Telo (3 X 3) | Fanoron-Dimy (5 X 5) | Fanoron-Tsivy(9 X 5)

What is your difficulty level?

Easy | Medium | Hard

Ready to start

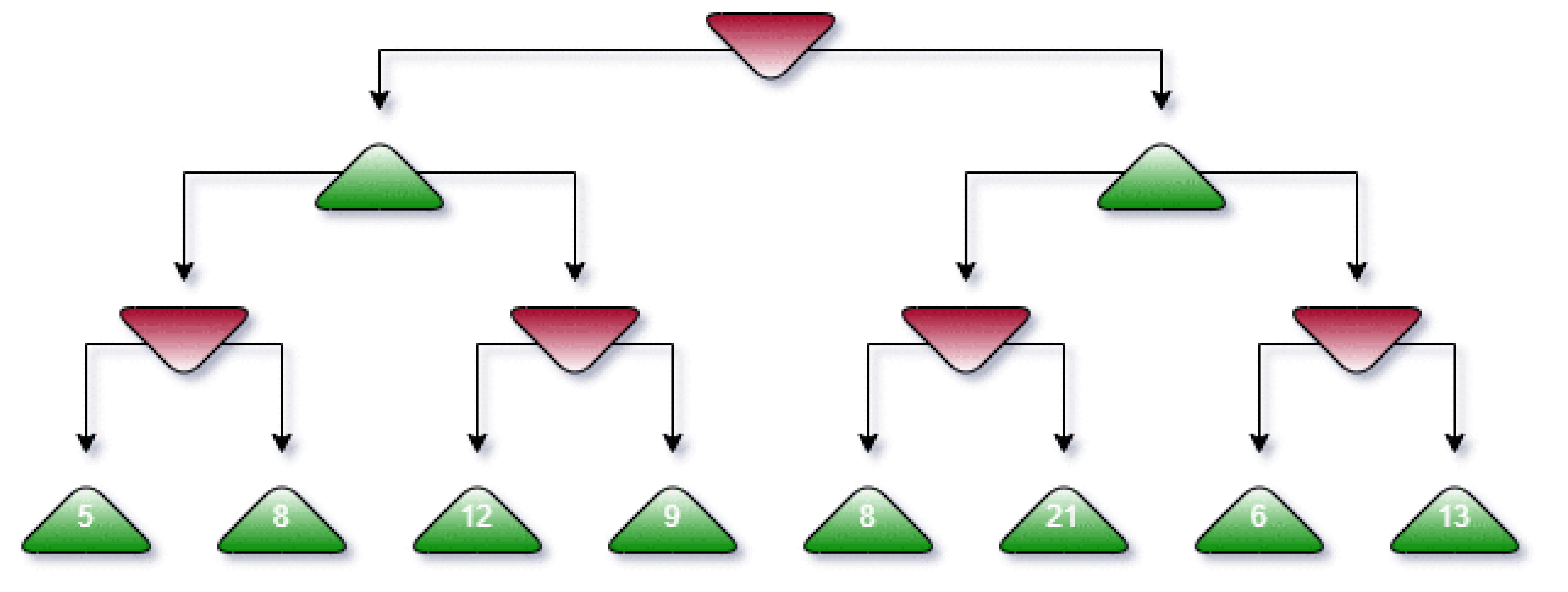
AI Agents

- Random
- MINIMAX
- MINIMAX w/ **αβ** cuts
- Monte Carlo Tree Search (MCTS)

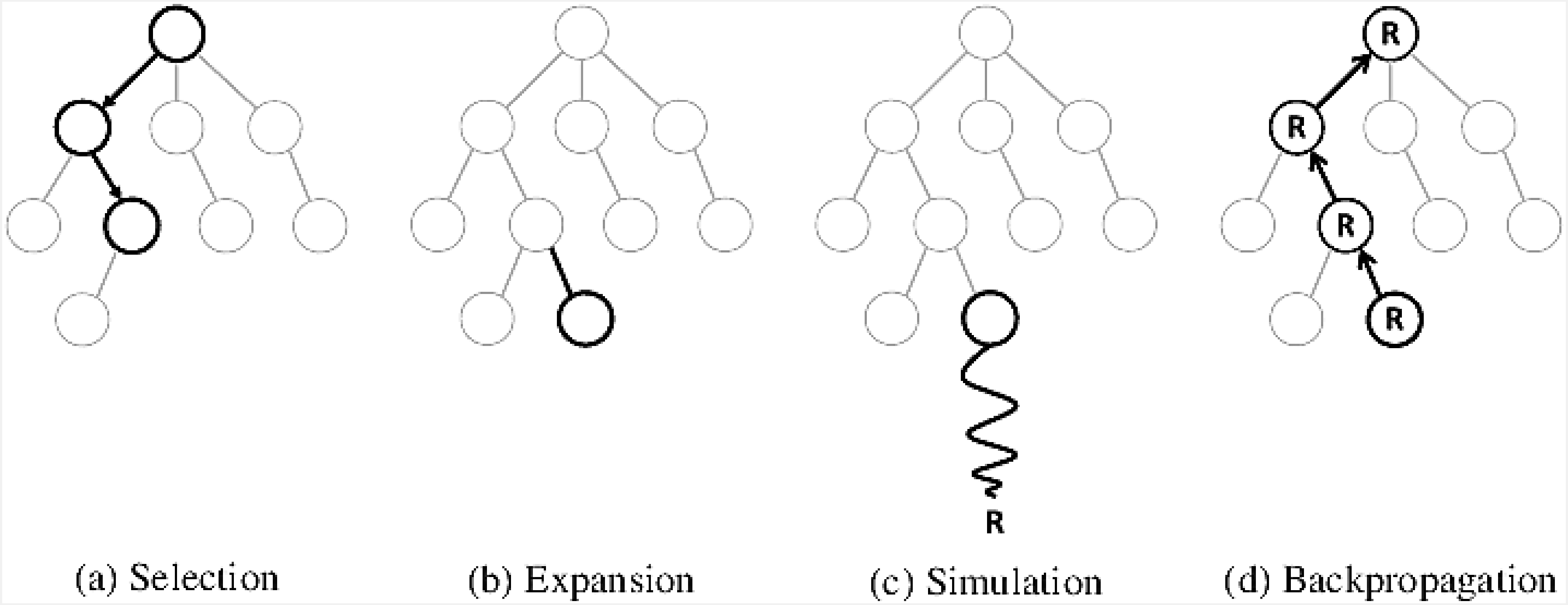
Utility based

MINIMAX (+ αβ)

- depth-first
- depth <-> difficulty
  - 3: Med
  - 5: Hard
- evaluation function
- αβ: greater efficiency



MCTS



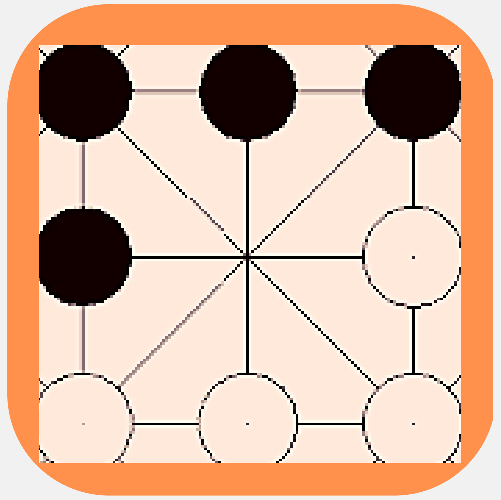
- Relevant Parameters
  - Max rollout depth
  - Nº iterations
  - C

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\textsc{Parent}(n))}{N(n)}}$$

Heuristic Evaluation Function

$$\frac{\text{N° of player tokens} - \text{N° of opponent tokens}}{\text{total N° of tokens}}$$

- For each token in a strong intersection +0.5 was added



- Evaluation function for MINIMAX and MINIMAX αβ, and biasing function for MCTS

Results

Measure computer performance in different **difficulty levels** (depth) and **board sizes**

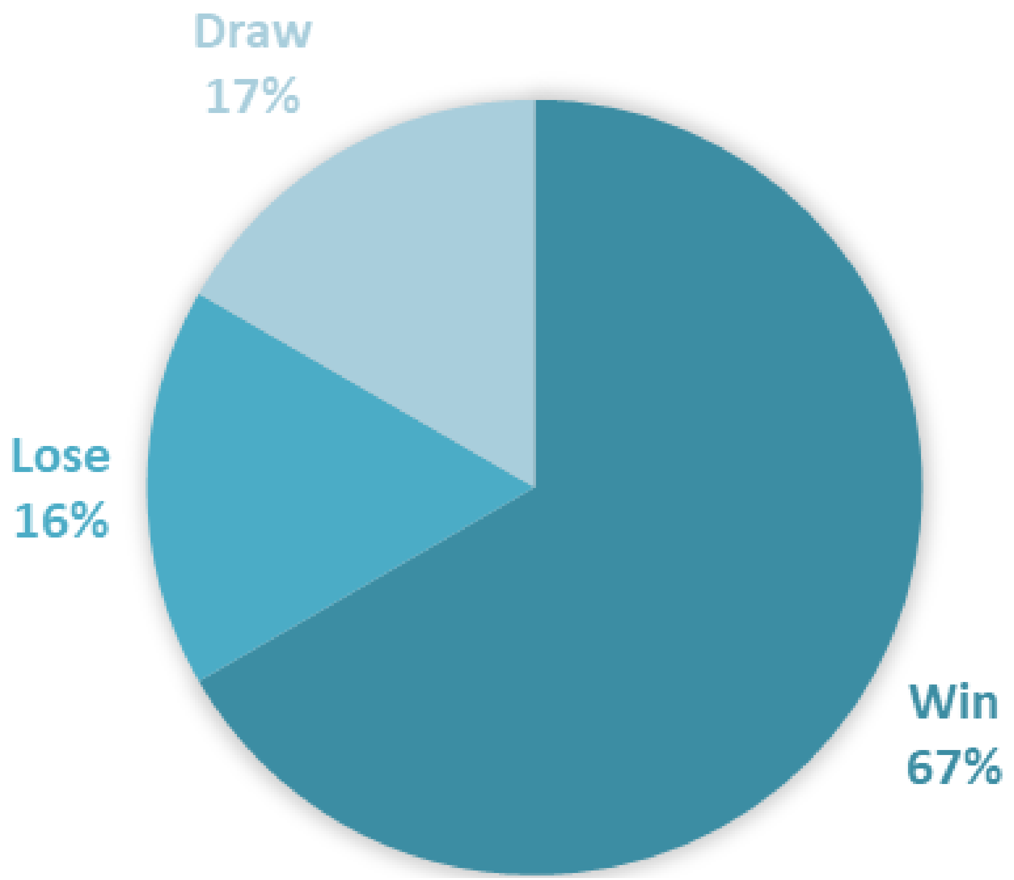
```
for size in [(3, 3), (5, 5), (9, 5)]:
    for level_1 in ['Easy', 'Medium', 'Hard']:
        for level_2 in ['Easy', 'Medium', 'Hard']:
            for alg_1 in ['Minimax', 'Minimax_AlphaBeta', 'Monte_Carlo_TS']:
                for alg_2 in ['Minimax', 'Minimax_AlphaBeta', 'Monte_Carlo_TS']:

                    # Initialize pygame
                    pygame.init()
```

- The effectiveness of the algorithms was measured by their **winning rate**
- AI vs random (for now)
- Game time duration
- Number of movements

AI vs Random

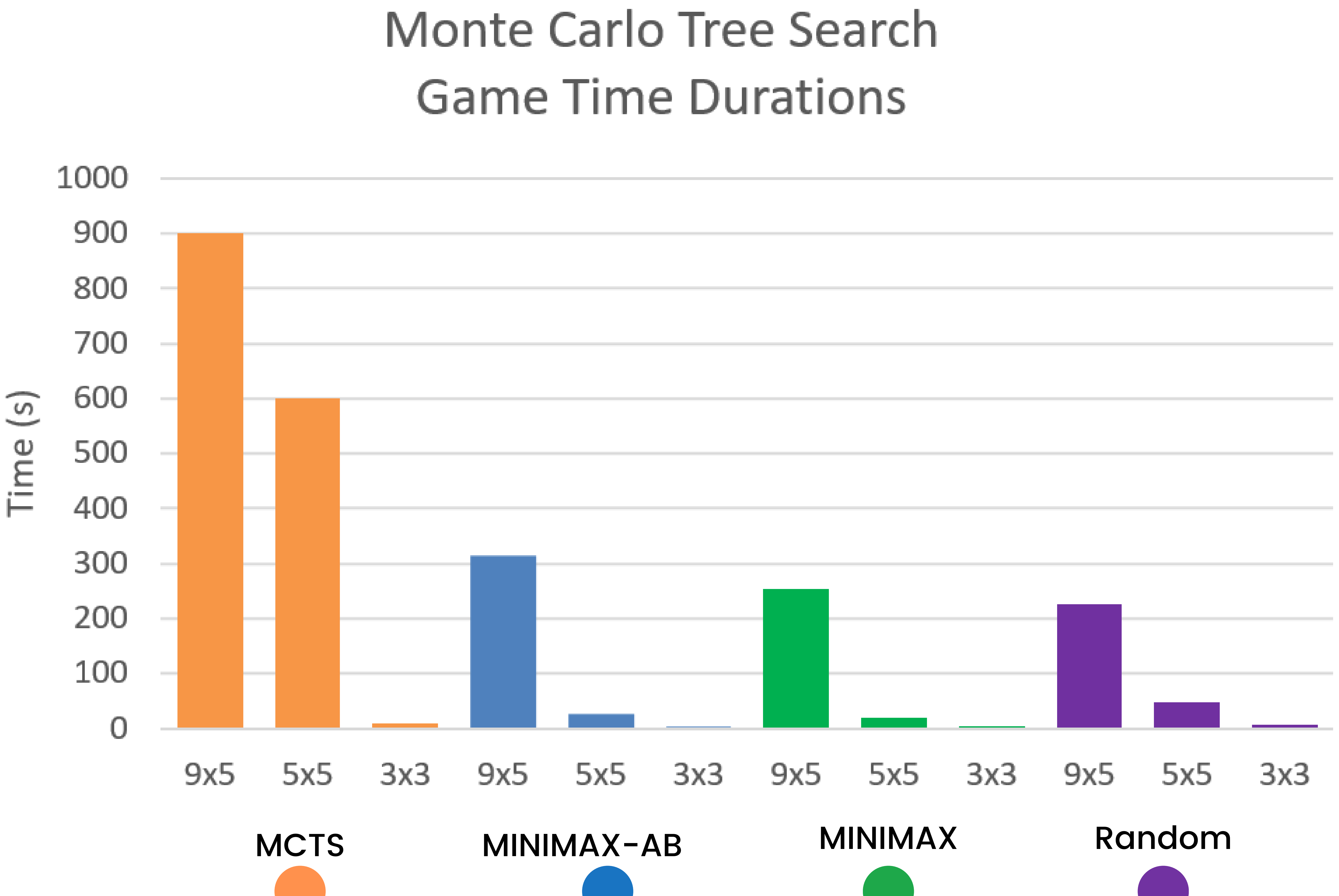
- MINIMAX wins 100%
- MINIMAX-αβ wins 100%
- MCTS wins 67%

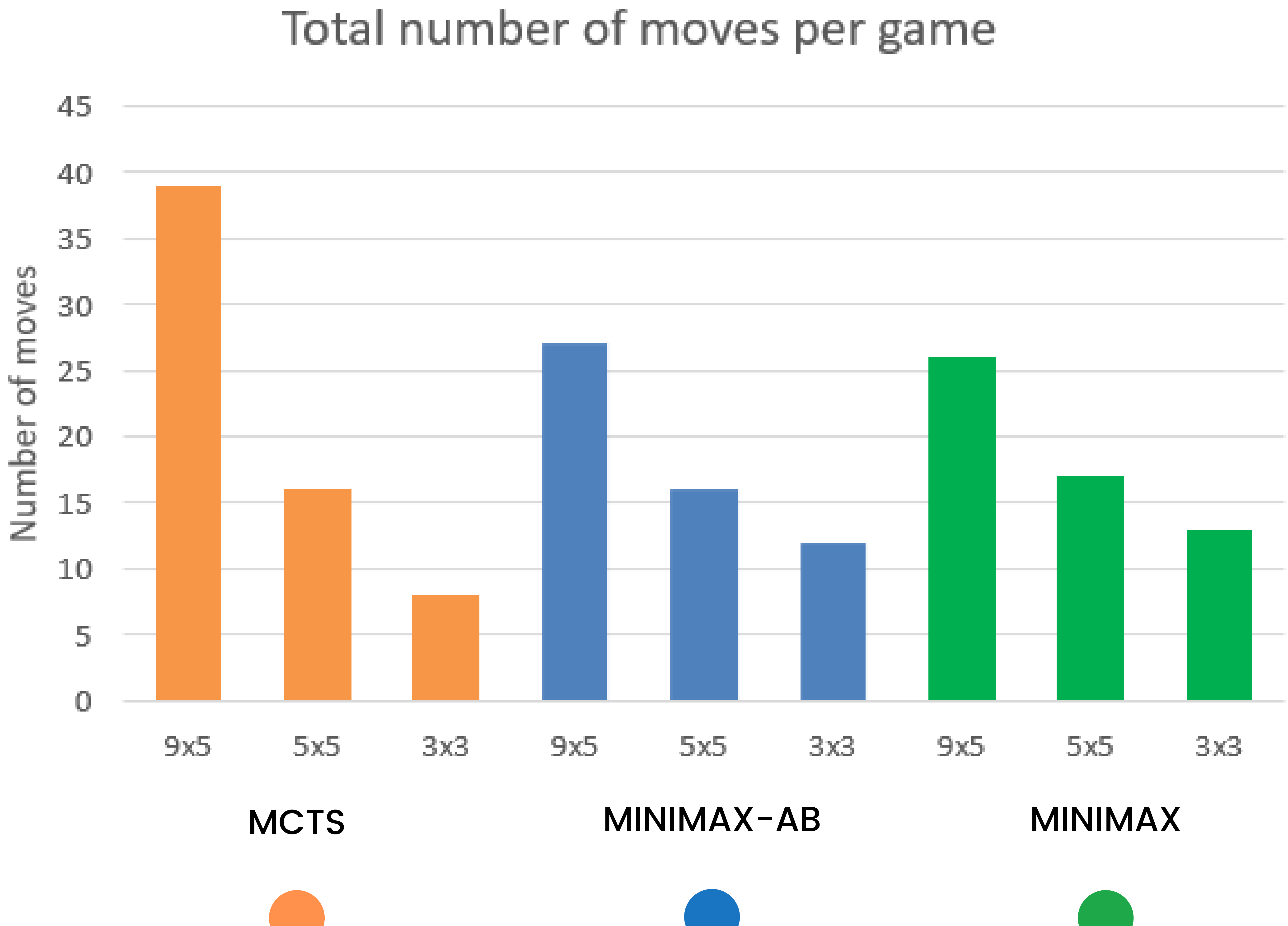


AI vs AI

5x5 Board

Algorithm	Match	Winner	Time	Number of moves
MINIMAX	hard vs medium	hard	3.2	18
MINIMAX w/ αβ		hard	1.6	18
MCTS		hard	52.6	20
MCTS hard	MINIMAX w/ αβ medium	MINIMAX w/ αβ medium	56.99	29


Monte Carlo Tree Search
Game Time Durations
Time (s)
1000
900
800
700
600
500
400
300
200
100
0
9x5
5x5
3x3
9x5
5x5
3x3
9x5
5x5
3x3
9x5
5x5
3x3
MCTS
MINIMAX-AB
MINIMAX
Random


Total number of moves per game
Number of moves
45
40
35
30
25
20
15
10
5
0
9x5
5x5
3x3
9x5
5x5
3x3
9x5
5x5
3x3
MCTS
MINIMAX-AB
MINIMAX

(Preliminary) Results

- All AI configurations win Random Agent.
- Monte Carlo is slower than all but can win with fewer moves - sometimes adapts an offensive attitude.
- Both MINIMAX algorithms have promising results.

Conclusions

<table>
<tr><th>Algorithm</th><th>Pros</th><th>Cons</th></tr>
<tr><td>MINIMAX</td><td>Guarantee to find optimal move</td><td rowspan="2">• computationally expensive
• prone to getting stuck in loops</td></tr>
<tr><td>MINIMAX w/ αβ</td><td>slightly faster than MINIMAX and much faster than MCTS</td></tr>
<tr><td>MCTS</td><td>can adapt a more offensive attitude</td><td>Takes much more time</td></tr>
</table>

Future works

- Allow player to make consecutive moves in one turn
- Add other draw situations
- Algorithm optimization: decrease space complexity
- MINIMAX: successor generation ordering
- MCTS: optimization of UCB and incorporation of draw condition in rollout phase