Tournament Results and Analysis

Introduction

This document presents the results of testing the MCTS (Monte Carlo Tree Search) algorithm. Note that the testing was constrained by hardware limitations, so further experimentation is encouraged for more comprehensive insights.

The UCT (Upper Confidence Bound for Trees) formula used in MCTS is as follows:

$$UCT = \frac{w_i}{n_i} + C \cdot \sqrt{\frac{\ln N}{n_i}}$$

where:

- w_i: Total reward of the child node.
- n_i: Number of visits to the child node.
- N: Number of visits to the parent node.
- C: Exploration parameter.

Experimental Parameters

The following parameters were used for the experiments:

- **max_iter** (int): Maximum number of iterations (used only if `cap_method` = 'iter').
- **C** (float): Exploration parameter for MCTS.
- **selection_method** (str): Method used for node selection ('random' or 'uct').
- **player_type** (str): Type of player ('MCTS' or 'random').

- **iterations_per_simulation** (int): Number of simulations per iteration.
- **max_runtime** (int): Maximum runtime allowed for the simulation (in seconds, used only if `cap_method` = 'time').
- **cap_method** (str): Method to cap the simulation ('iter' or 'time').

Tournament 1

For the first tournament, the following parameter configurations were tested:

```
params_list = [
    # Low exploration, single simulation per iteration
    {
        'max_iter': 10000,
        'C': 1.0,
        'selection_method': 'uct',
        'iterations_per_simulation': 1
    },

    # High exploration, single simulation per iteration
    {
        'max_iter': 10000,
        'C': None,
        'selection_method': 'random',
        'iterations_per_simulation': 1
    },

    # Random selection, single simulation
    {
        'max_iter': 1000,
        'C': None,
        'selection_method': 'random',
        'iterations_per_simulation': 10
    },

    # Multiple simulations per iteration
    {
        'max_iter': 15000,
        'C': 5.0,
```

```
        'selection_method': 'uct',
        'iterations_per_simulation': 50
    },

    # High exploration with multiple simulations
    {
        'max_iter': 20000,
        'C': 20.0,
        'selection_method': 'uct',
        'iterations_per_simulation': 100
    },

    # Random selection with multiple simulations
    {
        'max_iter': 25000,
        'C': None,
        'selection_method': 'random',
        'iterations_per_simulation': 50
    }
]
```

Analysis

A clear trend emerges: players using `uct` for node selection perform worse than those using random selection. This appears to stem from `uct`-based players focusing exclusively on attacking without employing defensive strategies. Experimenting with smaller values for the exploration parameter C (e.g., $C < 1$) might yield better results. For now, further testing will exclude the `uct` selection method.

Tournament 2

For the second tournament, the following parameter configurations were tested:

```
params_list = [
    {
        'selection_method': 'random',
        'iterations_per_simulation': 1,
        'max_runtime': 5,
        'cap_method': 'time'
```

```
    },

    {
        'selection_method': 'random',
        'iterations_per_simulation': 10,
        'max_runtime': 5,
        'cap_method': 'time'
    },

    {
        'selection_method': 'random',
        'iterations_per_simulation': 100,
        'max_runtime': 5,
        'cap_method': 'time'
    }
]
```

Analysis

These results are surprising. Increasing `iterations_per_simulation` beyond 1 reduced the AI's accuracy. This is likely because increasing iterations per simulation results in less uniform exploration of the game state. While more simulations per iteration should theoretically mitigate this issue over time, the 5-second runtime constraint appears to prevent these advantages from manifesting. For optimal performance within such constraints, conducting a single simulation per iteration seems preferable.

Results Format

Ten matches were conducted for each ordered pair of players. The results are presented in the following format:

- **Player 1 Wins**: A matrix showing how many matches (out of 10) Player 1 (row player) won.
- **Player 2 Wins**: A matrix showing how many matches (out of 10) Player 2 (column player) won.
- **Draws**: A matrix showing how many matches (out of 10) ended in a draw.

Results Visualization

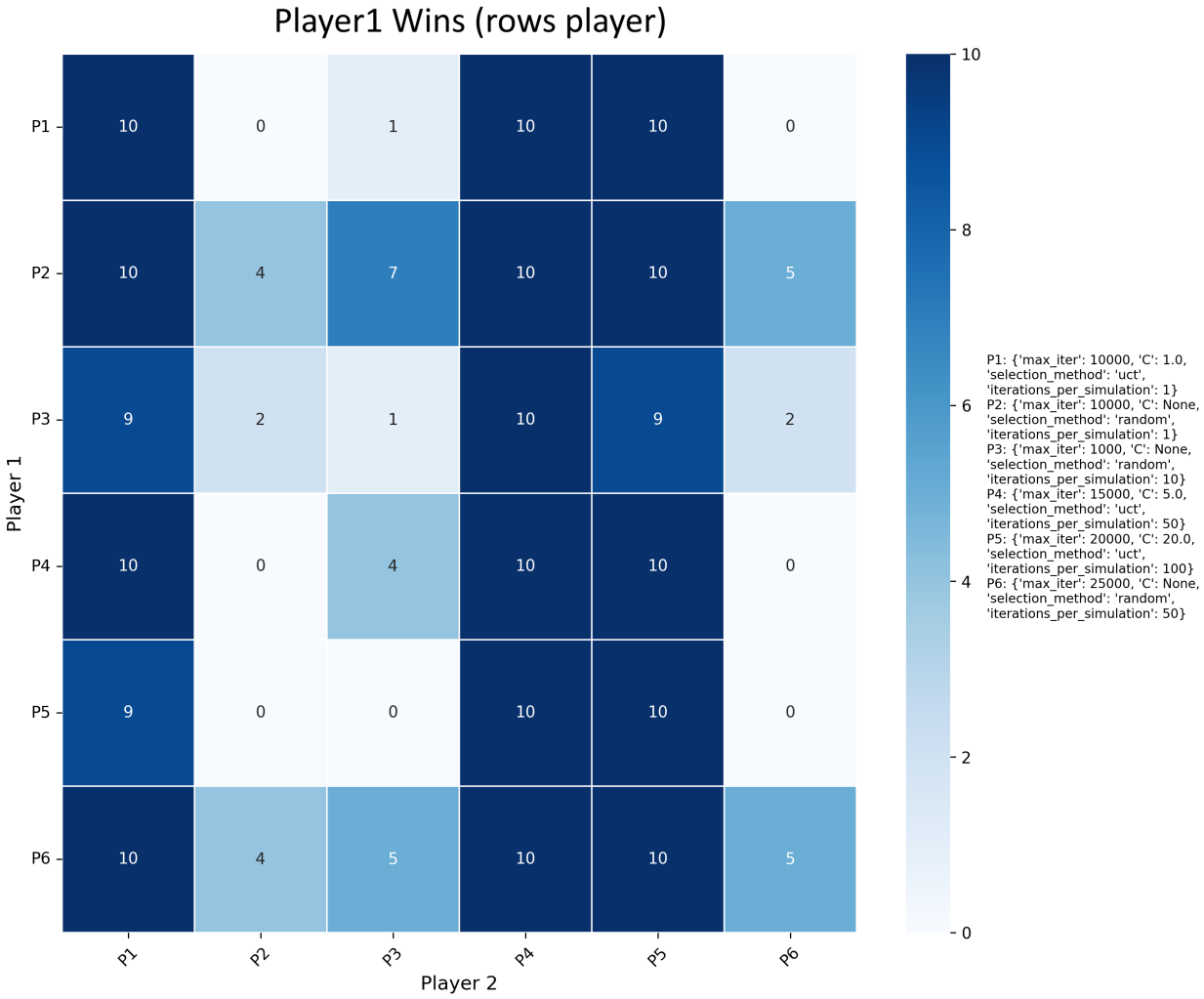

Figure 1: Results of Tournament 1: Player 1 wins.

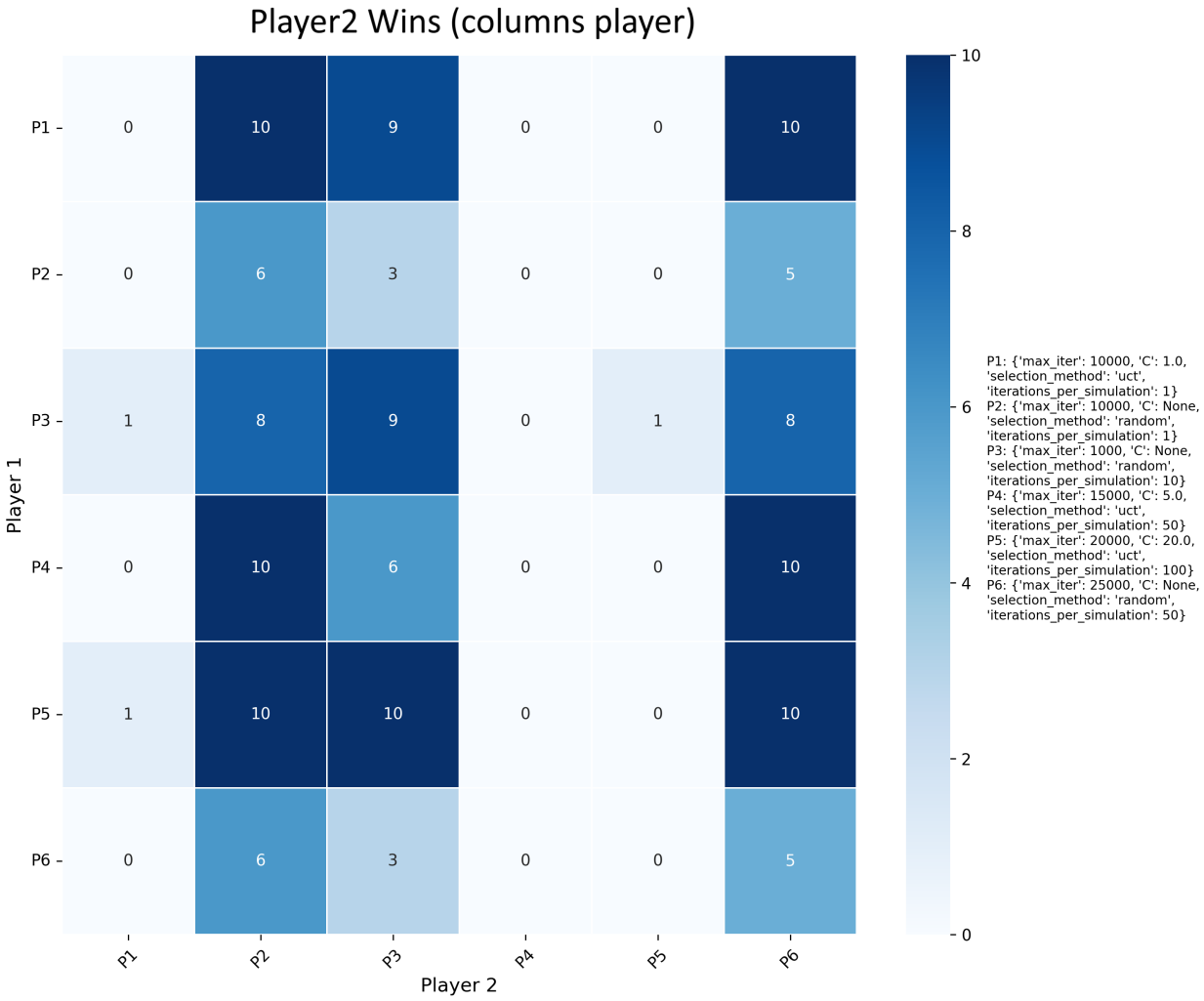

Figure 2: Results of Tournament 1: Player 2 wins.

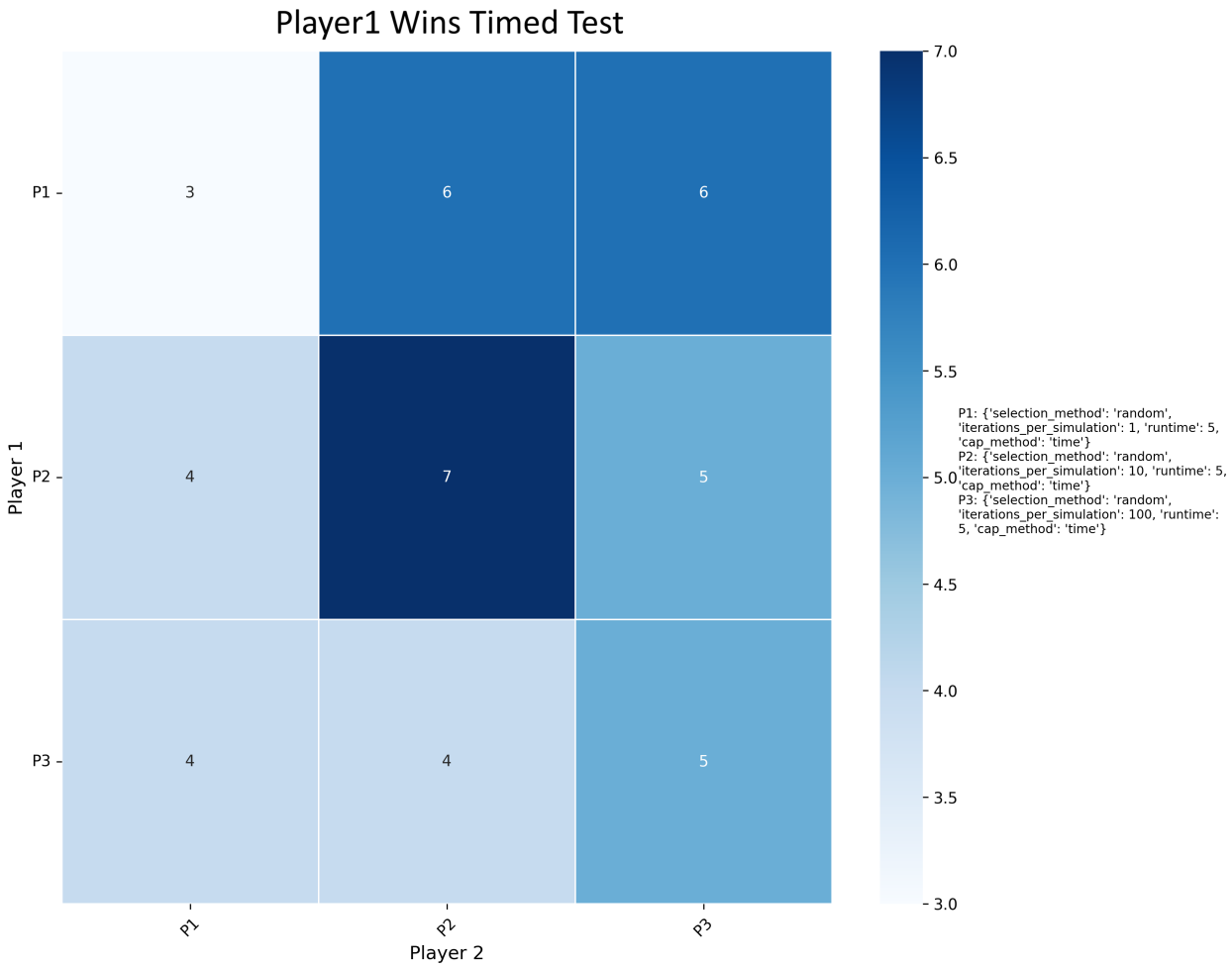

Figure 3: Results of Tournament 2: Player 1 wins.

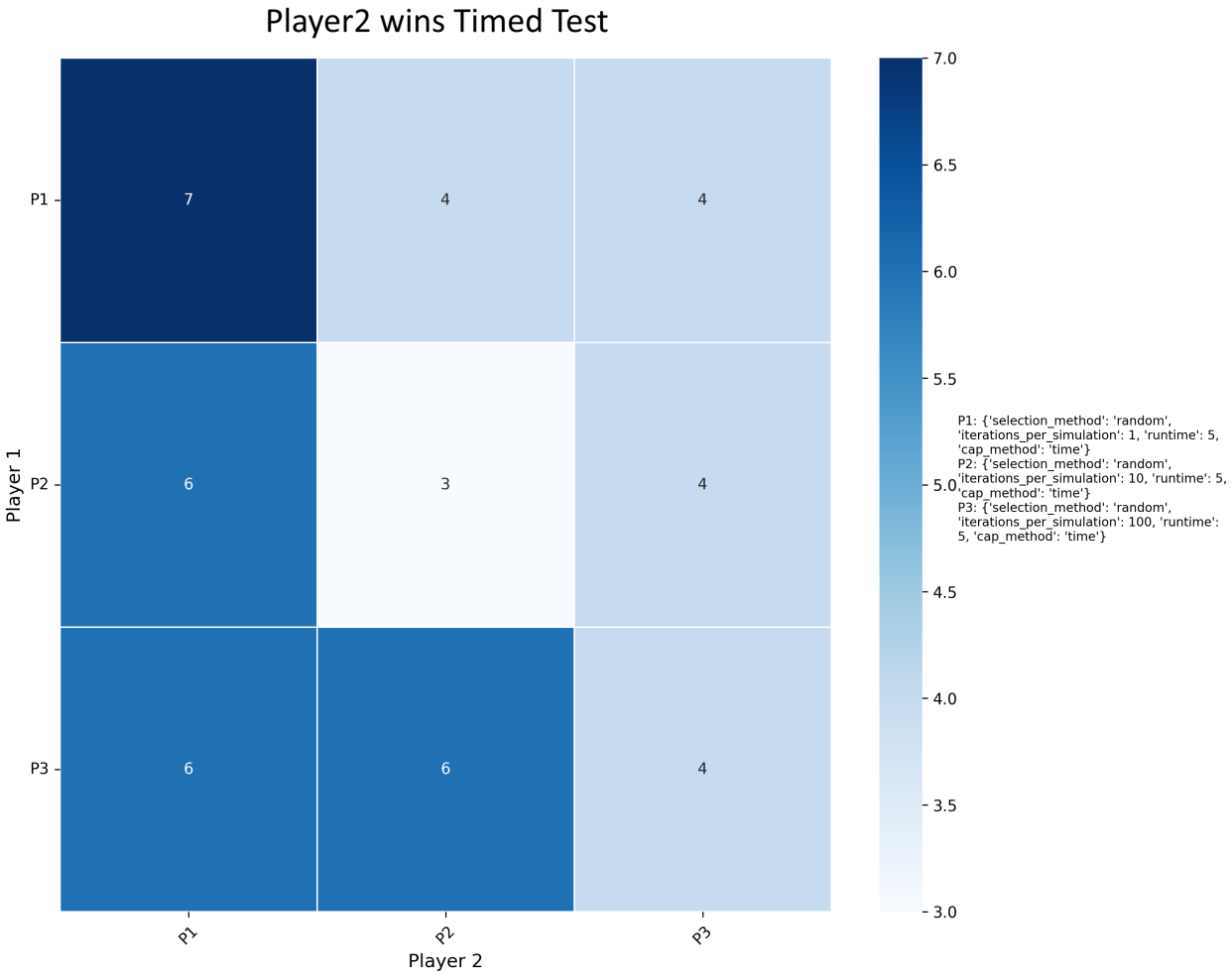

Figure 4: Results of Tournament 2: Player 2 wins.