Project 2 Report: Checkers-Playing Agents

John Brittain

COMS 4720

December 3, 2025

Introduction

For this project, I implemented three AI agents that play American Checkers:

1. **Alpha-Beta Pruning Agent**: Uses minimax with alpha-beta pruning and a custom evaluation function
2. **Monte Carlo Tree Search (MCTS) Agent**: Uses MCTS with UCB1 selection policy
3. **Hybrid Agent**: Randomly alternates between Alpha-Beta and MCTS for each move

This report includes observations of them.

Alpha-Beta Search Implementation

Evaluation Function

My evaluation function considers multiple factors beyond simple piece counting:

Features:

- Material count: Regular pieces = 1.0, Kings = 2.5
- Positional bonus: Pieces close to promotion get +0.3
- Defensive bonus: Pieces on back row get +0.2

The function returns a normalized score between -1 (bad for BLACK) and +1 (good for BLACK).

Search Depth Analysis

I tested the agent with search depths of 3, 5, and 7 ply.

Observations:

The AI beat me at every depth - I'm just not very good at checkers. However, I noticed clear differences in computation time. Depth 3 was noticeably faster, making moves almost instantly. Depth 5 had a brief pause before each move. Depth 7 took several seconds per move, especially in complex positions.

From a theoretical standpoint, deeper search should find better moves by looking further ahead. Depth 3 can only see 3 moves into the future, which might miss tactical sequences. Depth 7 can see much further and should play stronger, though there are diminishing returns - the difference between 5 and 7 is probably smaller than between 3 and 5.

For playing against a human, depth 3 seems optimal since it's fast enough to be responsive while still playing strongly.

Evaluation Function Comparison

I compared my improved evaluation function against a simple baseline that only counts material (pieces = 1, kings = 2).

Observations:

While I still lost games with both versions, I noticed the improved evaluation made more strategic moves. It seemed to:

- Keep pieces on the back row for defense longer
- Push pieces toward the opponent's side more deliberately
- Make better positional trades (giving up material for board control)

The simple baseline sometimes made moves that won material but gave up good positions. The improved version seemed to play more "human-like" by considering position alongside material.

Exploration Parameter Analysis

I tested different values of C in the UCB1 formula.

Observations:

With C = 1.0 (lower exploration), the agent seemed to commit to moves quickly. It found good moves but sometimes appeared to "tunnel vision" on one approach.

With C = √2 (theoretical optimum), the play seemed more balanced. It explored alternatives while still focusing on promising moves.

With C = 2.0 (higher exploration), the agent tried more diverse moves. It sometimes found creative solutions but occasionally spent too much time exploring weak options instead of capitalizing on strong positions.

The theoretical value of √2 seemed to work well in practice, providing a good balance between trying new things and exploiting what works.

Agent Comparison

Alpha-Beta vs MCTS

What I noticed:

Alpha-beta played very mechanically, repeated moves and near perfect execution.

MCTS played more unpredictably due to the random simulations. It seemed better at handling complex positions where tactics were less clear.

Alpha-Beta strengths:

- Deterministic and predictable
- Finds forcing sequences reliably

MCTS strengths:

- No hand-crafted evaluation needed
- Naturally balances trying new moves with exploiting good ones
- Computation scales linearly with iterations

MCTS weaknesses:

- Has randomness in move selection
- May not see forced tactics as clearly
- Needs many iterations for strong play

Hybrid Agent

The hybrid agent randomly switches between approaches each move.

Observations:

The hybrid agent's play felt inconsistent. Sometimes it would make a brilliant alpha-beta tactical move, then follow up with a more exploratory MCTS move that didn't capitalize on the advantage. The randomness didn't seem to provide any real benefit as consistency appeared more valuable than unpredictability in checkers.

Performance-wise, it seemed to average out to somewhere between the two individual agents.