WITS
UNIVERSITY

Reconnaissance Blind Chess Report

Group Members
Shayur Govin - 2558583
Rohan Chhika - 2543404

All our code can be accessed through Github: `GitHub Link`

May 21, 2025

Contents

1 Baseline Implementation(RandomSensing)

The *RandomSensing* agent was implemented per the project brief requirements as a baseline RBC Bot. It maintains a list of all possible board states throughout the game and updates this list using feedback from sensing and opponent moves. Sensing squares are chosen at random but they do avoid edge tiles to ensure full 3x3 windows. After each sense, inconsistent states are filtered out.

To select a move, the agent uses Stockfish to evaluate each possible state and applies a majority vote to determine the recommended move. To further refine the set of possible states, capture information and actual move results are also used. This agent satisfies all baseline requirements and is referred to as RandomSensing in the tournament comparison.

1.1 Results

Agent Results	**Wins**	**Losses**	**Draws**	**Win %**
RandomSensing	3	1	0	75.00
RandomBot	1	3	0	25.00
TroutBot	2	2	0	50.00

Table 1: Tournament results for `RandomSensing`. The agent performed well, winning 3 out of 4 matches and defeating both `RandomBot` and `TroutBot` once, indicating promising performance.

2 ImprovedAgent Implementation

The ImprovedAgent builds upon the baseline agent by enhancing the sensing strategy used and introducing a sequence of predefined opening moves to improve early game positioning, as detailed in the following sections:

2.1 Sensing Improvements

The first major improvement from the baseline agent was to change the `choose_sense` function. In the baseline agent, sensing was performed by randomly selecting a square on the board, without regard for what information might be gained. In contrast, the improved agent uses a knowledge-based approach to choose sensing squares more intelligently, with the goal of reducing uncertainty and identifying areas of importance.

The improved strategy begins by handling two special cases: if the list of possible states is empty, it defaults to selecting a square at random. If one of the agent's pieces was just captured, it prioritizes sensing the capture square, as this area is likely to contain the opponent's active pieces.

To improve efficiency, if the number of possible states is very large, a random sample of 1000 states is used. The agent then calculates the entropy for each square on the board. Squares where there is high disagreement about what piece is present have higher entropy. To give more weight to important pieces, entropy is scaled by the value of the piece (e.g., Queen > Pawn). Simultaneously, the agent evaluates threat frequency by checking how often each square is targeted by pseudo-legal moves across the possible states. This identifies which areas of the board the opponent is most likely to influence.

For each of the valid sensing squares, the agent scores its 3×3 window using a weighted combination of entropy and threat, according to the formula:

$$\text{score} = \alpha \times \text{entropy} + (1 - \alpha) \times \text{threat}$$

where $\alpha = 0.7$ to prioritize uncertainty over threat. The square with the highest combined score is selected as the sensing location. If no square stands out, the agent falls back on random choice.

This change makes sensing far more strategic. Rather than blindly guessing, the agent actively seeks areas that are both uncertain and potentially dangerous, helping it refine its belief about the game state and make better decisions in subsequent turns.

2.2 Predefined Moves

The second major improvement to the baseline agent was the introduction of predefined opening moves to serve as a guide to the agent in the early phase of the game. In the baseline agent, every move was selected using Stockfish evaluations. However, in the early game when uncertainty is low, this approach is computational and inconsistent.

To address this, the improved agent defines a hardcoded sequence of four aggressive opening moves for both white and black. These moves are designed to quickly advance a knight toward the opponent's king, with the goal of either pressuring or capturing the king.

During each of the first four turns, the agent checks if the next move in the sequence is currently legal and present in the list of valid actions. If it is, the move is executed and the agent progresses to the next move in the predefined sequence by incrementing an index. If the move is not available or the sequence is complete, the agent falls back to the Stockfish-based decision-making.

This makes the agent more efficient and structured during the opening phase while still retaining flexibility in the mid-game and late-game.

2.3 Results

Agent Results	**Wins**	**Losses**	**Draws**	**Win %**
ImprovedAgent	6	0	0	100.00
RandomSensing	2	4	0	33.33
RandomBot	3	3	0	50.00
TroutBot	1	5	0	16.67

Table 2: Tournament results for `ImprovedAgent`. The agent outperformed all other bots, achieving a 100% win rate against `RandomBot`, `TroutBot`, and `RandomSensing`, indicating that the sensing strategy and predefined move enhancements significantly improved performance.